Improvement of ECM Techniques through Implementation of a Genetic Algorithm

THESIS

James D. Townsend, Captain, USAF

AFIT/GE/ENG/08-34

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/08-34

Improvement of ECM Techniques through Implementation of a Genetic Algorithm

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

James D. Townsend, B.S.E.E.

Captain, USAF

March 2008

AFIT/GE/ENG/08-34

Improvement of ECM Techniques through Implementation of a Genetic Algorithm

James D. Townsend, B.S.E.E.
Captain, USAF

Approved:

Maj Michael A. Saville, PhD (Chairman)	4 Mar 2008 date
Dr. Richard K. Martin (Member)	4 Mar 2008 date
Dr. Seng M. Hong (Member)	4 March 2008 date

Abstract

This research effort develops the necessary interfaces between the radar signal processing components and an optimization routine, such as genetic algorithms, to develop Electronic Countermeasure (ECM) waveforms under a Hardware-in-the-Loop (HILS) architecture. The various ECM waveforms are stored in an ECM library, where an operator selects the desired function to use against a particular system. This optimization works with modular components, compared to previous research that embedded a genetic algorithm into the Range Gate Pull-off (RGPO) waveform optimization loop, which can be interchanged based upon the operator's desired hardware/ software testing setup. The ECM library's first entries contain the RGPO and Velocity Gate Pull-off (VGPO) signals, developed mathematically for multiple polynomial profiles representing realistic moving false targets. The Lab-Volt™ training system and jammer pod provided a validation medium for the developed RGPO and VGPO waveforms. These waveforms were optimized using a Simulink model of the Lab-Volt™ radar system and the MATLAB® Genetic Algorithm (GA) and Direct Search toolbox, contained in Version 7.4 (R2007a), using a defined parameter set, specified for the RGPO waveform. Integration of MATLAB® code with Simulink models provides the necessary interfaces to later transition from software radar models to actual system hardware. Results from GA optimization illuminate the necessity to specifically define the necessary constrains, both linear and nonlinear, imposed upon the environmental conditions. Given defined constraints relative to the Lab-Volt™ training system, the HILS architecture produced multiple constant velocity range profiles with walk-off ranges and maximum velocities similar to the Lab-Volt™ Jammer Pod.

Acknowledgements

First and foremost, I would like to thank my advisor, Maj. Michael Saville (AFIT/ENG) for the constant motivation and support during these past few months. His continuing persistence towards small project details allowed me to stay focused throughout. This project could not have been possible without the funding and support from Dr. Seng Hong (AFRL/RYRA) and the Air Force Office of Special Research. To the NASIC Radar Team, especially Bob Russell, thank you for the introductory radar analysis training and inspiration to pursue an advanced academic degree in radar systems. Mr. Greg Taylor's intimate knowledge of the wideband equipment allowed for better understanding of the Lab-Volt™ radar system. Greg also served as a knowledgable sounding board to work through rough spots in my research, giving wonderful insight from an outside vantage point. Capt. Adam Liddle helped keep the RADAR lab motivated with his "soap-box sermons" and acting as a sounding board for numerous writing/research questions. Finally, I would like to thank my wife for all the love and support throughout my research. No matter whether research was going well or hitting infinite snags, she was there to keep my batteries charged and constantly remind me of my capabilities.

James D. Townsend

Table of Contents

Page

Page

List of Figures

List of Tables

List of Symbols

Symbol		Page
$\mathcal{K}$	ECM Scoring Function	26
$\Phi(u)$	RGPO Signal Operator	26
$\mathcal{L}[\Phi(u)]$	Radar Signal Processor Operator	26
M_{J}	Number of CPIs in Walk-off Time	31
R	range	37
t	time	37
c	speed of light, 3x10$^8 \frac{m}{s}$	37
τ	Pulse Width	37
$r(t)$	Aircraft Position	38
$v(t)$	Aircraft Velocity	38
$a(t)$	Aircraft Acceleration	38
f	RGPO Polynomial Factor	39
R_o	False target initial delay	39
R_{max}	False target maximum delay	39
T_w	False target walk-off time	39
T_{PRI}	Processing Time	40
V_{max}	Maximum Range Rate	42
V_{min}	Minimum range rate	44
A_{max}	Maximum Acceleration	46
A_{min}	Minimum Acceleration	48
P_t	Transmitted Power	52
$A_{e_{\text{jam}}}$	Jammer Effective Receive Area	52
P_O	Signal Power at Target	52
P_R	Received Signal Power	52
V_r	Target Received Voltage Signal	52
Ψ_0	Mean Noise Power	52
V_e	Expected Detection Voltage	52
G_{JR}	Repeater Jammer Gain	53

Symbol		Page
L_p	Polarization Loss	53
G_{JT}	Jammer Transmitter Gain	53
G_e	Repeater Overall Gain	53
σ_e	Repeater RCS	53
G_{REP}	Total Repeater Gain	53
A_{j_o}	Initial Jammer Signal Amplitude	55
$A_{j_{\min}}$	Minimum Jammer Signal Amplitude	55
δ_i	Initial Delay	58
A_j	Jammer Signal Voltage	61
ω_J	Normalized Doppler Frequency	61
ϕ_J	Jammer-Induced Phase Shift	61
b_J	Jammer Phase Bits	62
v_d	target Doppler velocity	65
q	STAP sensor number	66
$\overline{v}$	STAP steering vector	66
$\overline{b}(t)$	Temporal Steering Vector	66
$\overline{a}(t)$	Azimuth Steering Vector	66
$\overline{e}(t)$	Elevation Steering Vector	66
(μ_x, μ_y)	Sample Mean	82
(ϵ_x, ϵ_y)	Sample Error	82
(σ_x, σ_y)	Sample Set Standard Deviation	82

List of Abbreviations

Improvement of ECM Techniques through Implementation of a Genetic Algorithm

I. Introduction

Successful development of electronic countermeasure (ECM) techniques against target tracking radars comes through extensive analysis of hardware system implementation. Currently, this ECM development and analysis takes numerous manhours to complete and can be a financially expensive venture. Numerous resources are allocated in the setup and investigation/evaluation of a system's circuit design. Those resources do not cover any required for selection criteria determined for the types of ECM to run against a given platform [1]. This research, made possible by the Air Force Office of Scientific Research funding the basic research, continues an ongoing effort involving the following organizations: Air Force Research Laboratory, RF Sensor Technology Division, RF Countermesaures Assessment Lab (AFRL/RYRA); Air Force Research Laboratory, Information Technology Directorate, Embedded Technology Systems Engineering (AFRL/IFTA); Air Force Institute of Technology, Electrical and Computer Engineering Department (AFIT/ENG); and research support from in-house contractors. This collaboration is focused on implementing a hardware-in-the-loop simulation (HILS) for developing ECM techniques using a genetic algorithm. This research investigates the necessary linkages between radar signal processing hardware and the genetic algorithm (GA) optimization routine to produce the desired Range-Gate Pull-Off (RGPO) or Velocity-Gate Pull-Off (VGPO) ECM techniques. This system implementation emphasizes the necessity to reduce ECM technique development time and develop a library of fitness functions that can be used against multiple radar systems in a multitude of engagement situations.

The purpose of this chapter is to outline the efforts of this research focused on integrating hardware equipment, such as the Textronix RSA6114A Realtime Spectrum Analyzer, with the genetic algorithm, encoded in MATLAB® V7.4 (R2007a) and hosted on the Textronix AWG7102 Arbitrary Waveform Generator, in support of optimizing ECM waveforms against threat hardware. The search space landscape exploited by the genetic algorithm portrays the threat hardware's operational environment, which can be optimized through understanding the mathematical models representing the environment. An ECM development suite is designed to handle real-world applications with basic understanding of the connections between test and operational radar systems. This thesis develops the process necessary to develop these interactions and enable a HILS architecture for ECM waveform development as part of the technique generation, optimization algorithms, and objective functions that must be defined in a general sense. This chapter addresses the background, problem to be investigated, and the proposed methodology.

1.1 Background

ECM development falls directly under the electromagnetic jamming component of electronic attack (EA) in support of the tenets of electronic warfare (EW). According to Air Force Doctrine 2-5.1, "Electromagnetic jamming is the deliberate radiation, reradiation, or reflection of electromagnetic energy for the purpose of preventing or reducing an [threat system's] effective use of the electromagnetic spectrum, with the intent of degrading or neutralizing the [threat system's] combat capability" [2]. Figure 1.1 depicts the EW arena with its major components. The cooperative aircraft are shown in the center. The ground radar and airborne guidance system work in concert with the tracking radar to provide guidance information. The dashed lines depict the ECM signals used to deceive the tracking systems and disrupt, degrade, or deny lock-on and accurate target guidance. ECM jamming signals limit the threat's access to information on cooperative force movement and composition.

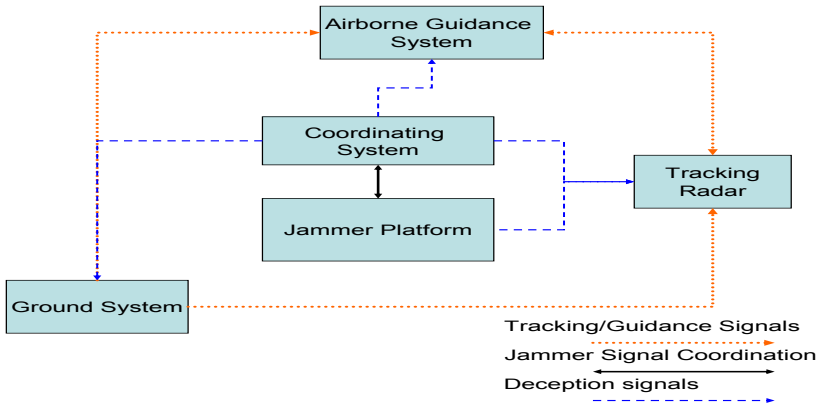


Figure 1.1: Electronic warfare environment

Deception jammers are a specific type of jammer sensitive to technological advances and constant environmental changes. This specific jammer type varies from the common jammer conception of active noise jamming to highly sophisticated waveforms for deception. Deception jammers do not overwhelm the EM spectrum with external noise, but provide false ranging information to the victim system. Deception examples include Range Gate Pull-Off (RGPO), where pulse returns are time-delayed to induce an increase in target distance, and Velocity Gate Pull-Off (VGPO), which modulates pulse returns to increase measured velocity readings. If the deception jammer provides accurate false target information, the victim system becomes confused between real and false data and can no longer extract valid targeting data [3]. By nature, deception jammers are more sophisticated than noise jammers, requiring more complex hardware and software to create the desired signals. Deception jammers are more complex because operational and waveform parameters are directly correlated to the victim system's performance parameters and modes of operation. Developing ECM waveforms for use against search, acquisition, and tracking systems requires having the actual hardware available for testing and exploitation. This task becomes problematic because every radar system can not be acquired, which dictates that other methods are necessary to develop ECM techniques. The question then arises: If adversary assets and testing ranges are in high-demand, how can the process of ECM waveform generation be automated and optimized to reduce technique development

time? This project focuses on developing a generic hardware-in-the-loop simulation system consisting of: an automated optimization algorithm, like GA, a library of fitness functions for ECM techniques, and a radar/threat environment. Before discussing the proposed methodology, the Lab-Volt™ radar training system is presented as the test radar.

1.1.1 Lab-Volt™ Radar Training System. For modeling and implementation, the Lab-Volt™ system serves as a demonstration platform for understanding how tracking radars work in a simulated environment. The Lab-Volt™ radar is a laboratory-scaled system with the capability to conduct both range and angle tracking of a single point-scatterer at distances of up to 7.2 meters. This system has numerous variable settings for operational frequency, pulse repetition frequency (PRF), and pulse width to enable investigation of signal variation as encountered in real-world systems. Table 1.1 lists Lab-Volt™ configuration data for implementing the Target Tracking mode. Understanding the Lab-Volt™ radar tracking system reveals how real-world considerations are modeled with validity. The Lab-Volt™ radar consists of modular reconfigurable components for range and angle tracking, moving target indicating, Doppler processing, and clutter simulation. The Lab-Volt™ Target Tracker Module uses the Moving Target Indicator (MTI) output to detect objects in the environment. Range and angular data received from the dual-feed parabolic-reflector antenna are determined to resolve the target's movement. Depending on the resulting error signal received from the monopulse antenna, power signals can be sent to the Antenna Motor Driver to compensate for the target's positional changes. Angu-

Table 1.1: Lab-Volt™ Tracking Radar Parameters [4]

Parameter	Values
PRF f_{prf}, Hz	12, 18, 144, 216, 288
PRF Modes	SINGLE, STAGGERED
Pulse Width τ, (ns)	1-5
Operating Frequency Range f_t, (GHz)	8.0-10.0
Nominal Operating Frequency $f_{t_{nom}}$, (GHz)	9.4
Range Distances (m)	1.8, 3.6, 7.2

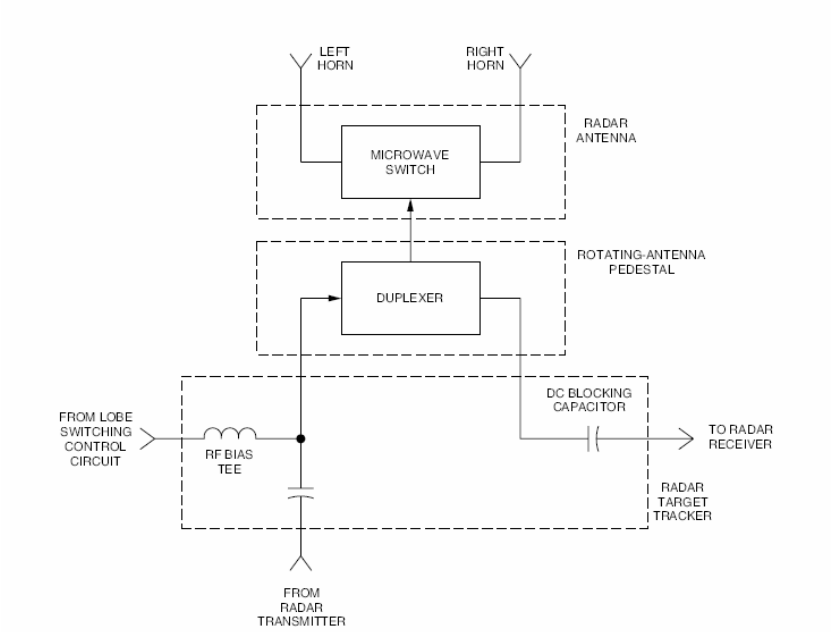


Figure 1.2: Lab-Volt™ dual-channel antenna RF setup [4]

lar compensation by the Target Tracker module allows for accurate tracking profiles by the Lab-Volt™ system. Automatic angle tracking within the Lab-Volt™ system allows for the tracking of moving targets with automatic alignment of the aperture to the target's location [4]. Figure 1.2 shows a circuit diagram of the dual-channel aperture. The output received from the left and right channels of the dual-channel aperture are compared to determine angular direction. The MTI video signal is compared to the range gate information and converted into an inverted (Right Lobe) and non-inverted (Left Lobe) signal. Signal splitting occurs in this fashion due to the aperture geometry. The antenna is configured to return the error between the two horns, where the zero error ground reference indicates target position along the center-line, and a positive-valued signal represents a target coming from the antenna's left. While the MTI processor does not distinguish where the horn references from in creating the video signal, a gate timing circuit differentiates the left and right lobes from the inverted signal. A resulting positive signal then means that the target is left of boresight, causing an azimuth correction to be sent to the Antenna Motor Driver to move left (clockwise) of its current position. This movement decreases angular error to maintain a successful angular lock. The Lab-Volt™ system then can model appro-

priately scaled 2-D scenarios for tracking targets in range and azimuth. Furthermore, limited Electronic Counter-Countermeasures (ECCM) tracking capabilities such as leading-edge tracking add realism for modeling the electronic warfare environment.

1.1.2 Lab-Volt™ ECCM Capabilities. The Lab-Volt™ system allows for accurate real-world modeling through the ECCM capabilities built into the range-azimuth tracking system features. The first involves the leading-edge tracker. Implementation for this tracking method involves moving the target tracking point off the pulse centroid and to a specific point on the pulse's leading edge. During the late gate portion of the range tracking cycle, a DC voltage is held instead of the late gate signal. The held DC voltage causes a bias in the integration circuit, subsequently moving the track point toward the leading edge. The leading edge tracker threshold dial on the Target Tracking module adjusts the DC levels accepted by the tracking algorithm. While using the leading edge function of this Lab-Volt™ component makes locking onto quick-changing target returns more difficult, the benefit comes in preventing range gate detection by jamming systems. The leading edge tracking contained within the Target Tracking module prevents detection of false targets intended to mask the original target and change the range gate information. This technique prevents range-gate pull off jamming from effectively walking the tracker's gated range value off the actual target return. Another important tracker ECCM feature involves the detection of average range gate limiting. The target's average rate of change detected by the radar is shown in Figure 1.3. Engaging this tracking radar feature enables detection of unnatural range gate changes. Jamming techniques, such as those that delay the target return in time, cause a tracker to see this movement and estimate appropriate changes to the false target return. These erroneous estimations cause the radar to lose track because the range gate no longer represents the location where the target is located. The average rate limiter calculates the range gate between collected returns and rejects the gate changes that exceed the determined threshold. The rate limiting signal shown in Figure 1.3 determines the maximum change possible

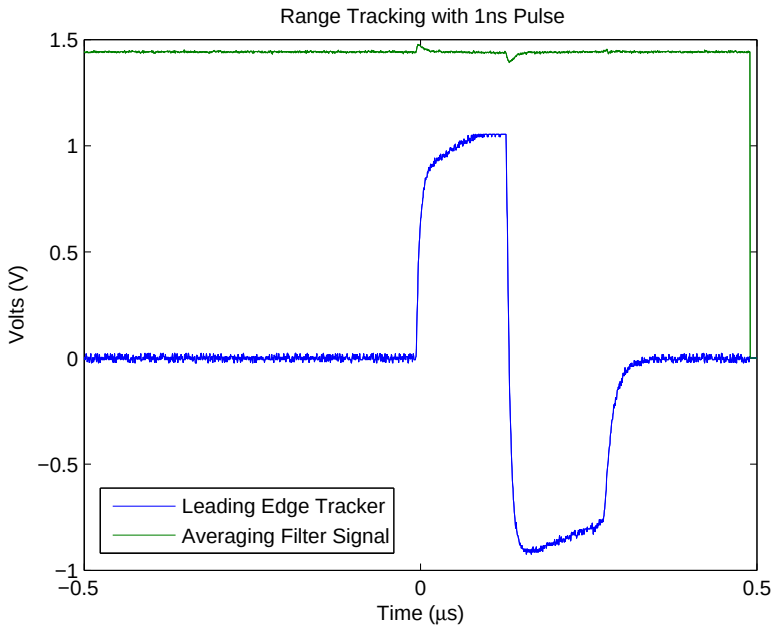


Figure 1.3: Averaging filter Lab-Volt™ signals [4]

for the given range recorded from the rangle/angle tracking algorithms as the range rate limiter. Velocity and Range Gate Pull-Off jamming signals must compensate for certain system restrictions, like range-limiting, to execute target masking without alerting the victim system of a false target [3]. These two ECCM safeguards allow for real-world threat modeling of tracking systems with appropriate considerations of modern systems.

1.1.3 Summary. The Lab-Volt™ training system serves as a characteristic system for developing ECM techniques. While limited to only 2-D tracking capabilities, this system serves as a representative piece of hardware used by operators to understand and become proficient with real-world systems. Furthermore, the Lab-Volt™ tracking module provides ECCM techniques, similar to those used in the operational environment, that must be overcome in order to properly implement ECM waveforms. The necessary tracking signals can be exploited to develop a ECM technique generator that can be optimized by a genetic algorithm. It is the intent of this research to further progress HILS development that creates and evaluates ECM tech-

niques using automated optimization. Having an operational HILS system that can optimize ECM waveforms requires a library of objective, or fitness functions. These functions should not be integrated into the optimization such that a new optimization algorithm must be created for each combination of technique optimization.

1.2 Research Problem

Current ECM technique development is accomplished through numerous trial and error iterations that are time-consuming and require access to test assets and a priori knowledge of the system. Furthermore, during the development process, human error can play a significant role in the time necessary to exactly characterize the desired waveform against a specific asset. Genetic algorithms have the potential to reduce development time through effective ECM waveform and technique derivation [1]. In an effort to develop effective ECM, this project aids in the development of a generic HILS system [5]. Through developing HILS with onboard optimization, human error may be reduced and development time can be reduced because computerized optimization is more precise and faster than solely man-in-the-loop experimentation.

This research answers the question: What interfaces are required between the radar signal processing components and the GA to develop ECM waveforms under HILS? Three major topics must be addressed. The first is Lab-Volt™ operation for range and angle tracking. The second is a thorough understanding of how the radar signal processing suite will be used to determine if the victim radar has a successful track. The final aspect is the mathematical modeling of ECM waveforms for waveform generalization and optimization.

1.3 Scope/Methodology

This research builds upon recent Lab-Volt™ characterization by Mayhew [6] and extends it to the tracking module as a benchmark for understanding real-world tracking radars. By understanding how the Lab-Volt™ tracks targets, the HILS systems can be demonstrated with a GA optimization. The primary contribution

of this work is the architecture for including a library of fitness functions for ECM waveform generation. In addressing the research question and the investigative tasks from the previous section, the following questions must be addressed:

1. Lab-Volt™ Tracking Module
   (a) How does the tracking module implement range/angle tracking?
   (b) What defines the tracking module "breaking lock"?
   (c) How exactly are ECCM techniques implemented that are inherent to the tracking module?
2. Radar Signal Processor
   (a) What determines that a RADAR is tracking a given target?
   (b) What determines the best ECM technique to use against that RADAR?
   (c) What determines if the "break lock" criteria are met?
3. Genetic Algorithm
   (a) What is the fitness function?
   (b) How should the GA search space be defined and what is the space?
   (c) What physical limitations (i.e. design variables and constraints) need to be represented in the GA search space?
   (d) Will a multiple-objective GA be required for the best ECM technique selection?
   (e) What interfaces are required between the calculated GA results and the Arbitrary Waveform Generator (AWG)?

These questions provide a road map through the necessary tasks needed to successfully integrate optimization with HILS. The first set of questions regarding the radar tracking module requires development of a tracking module in MATLAB® for the

current radar model. Previous research implemented the transmission and radar receiver portions of the Lab-Volt™ system but did not include the tracking portion. Range and angle trackers and their ECCM are studied to answer the first set of questions. The second group of questions addresses the need for developing mathematical models of the RGPO/VGPO techniques. The ECM models must be more generalized than the RGPO model reported by Nunez et al. [1]. Understanding the Doppler responses implemented in a VGPO jammer reveals how a space-time adaptive processing (STAP) model of the ECM Jammer pod may accelerate the optimization. The MATLAB® model of the ECM Jammer pod subsequently establishes how the radar signal processing equipment determines if the threat radar broke lock. The final group of questions explores the GA and the equations necessary to represent the landscape. In exploring the VGPO and RGPO jamming techniques, exploration focuses on how to use STAP data simulation methods to define the fitness function. While the background research explores how GAs can assist the problem, the requirements expand for multi-optimization when multiple independent optimization routines are running simultaneously.

Resolving these questions will drive the HILS development including the GA implementation stage into exploration of multiple jammers used in concert with each other. While this system is being developed on a training radar system, the next logical step is to test the HILS system on a real-world system. Accurate modeling of the tracking radar will give a robust development of the testing hardware and software such that the system accurately interacts with systems that operate at different operational frequency bands and varying pulse widths or PRFs in a variety of methods. The methodology of how this research will progress to through HILS development will be discussed further in chapters three, four, and five.

II. Tracking Radar and Genetic Algorithm Literature Review

2.1 Chapter Overview

This chapter discusses the background material necessary for understanding the reasoning behind choosing the Genetic Algorithm optimization technique for developing ECM techniques against tracking radar systems. As discussed in the previous section, the Lab-Volt™ system serves as a laboratory-scaled system suitable for conducting ECM waveform optimization. Tracking radar implementation and genetic algorithm development are presented in this chapter to give the reader a better understanding of how the ECM waveform optimization will be implemented against the Lab-Volt™ system. Section 2.2 covers tracking radar principles used by the Lab-Volt™ system. Section 2.3 gives an overview of genetic algorithms, outlining the basic concepts and principles used to optimize the ECM technique fitness function. Finally, Section 2.4 gives a literary overview of how genetic algorithms and evolutionary computational methods have been used to solve similar optimization problems.

2.2 Radar Tracking Model

2.2.1 Radar Range Equation. Development of radar tracking techniques derives from the radar range equation studied for a single pulse return. This single return is then used for detecting targets at a desired range. Equation (2.1) is the basic radar range equation, applying the situation where a continuous track is held for a given time, t_o [7]. Equation (2.1) includes effective aperture (A_e), transmit antenna gain (G_t), target cross-section (σ_t), and receiver noise figure ($k_B T_o B F_n$) as the detection of targets in range. In Equation (2.1), the product of average power returned (P_{av}) and the time on target must equal the transmitted energy (E_t) broadcast from the transmitter. The signal-to-noise ratio (SNR), denoted as (S/N_o), is defined as the ratio of signal power collected at the receiver to the noise power. N_o is the height of the flat noise power spectral density. Equation (2.1) determines the maximum range

at which a target of RCS σ can be "seen" over a given period t_o.

$$R_{\max}^4 = \frac{P_{av} t_o G_t A_e \sigma}{4\pi k_B T_o B F_n (S/N_o)} \tag{2.1}$$

The transmit antenna gain is

$$G = \frac{4\pi \rho_a A}{\lambda^2}, \tag{2.2}$$

where A is the antenna's physical area and ρ_a is the antenna efficiency defined in [8]. Antenna gain is derived from the radiation intensity ($U(\theta, \phi)$):

$$G = \frac{4\pi U(\theta, \phi)}{P_{in}}. \tag{2.3}$$

The directive gain in $U(\theta, \phi)$ is expressed as the maximum radiation intensity along each field component [8]. Angular accuracy is based on the resolution of θ_B and ϕ_B respectively, where θ_B and ϕ_b are the 3dB beamwidths in azimuth and elevation respectively [9]. The Lab-Volt™ gain of 28.9dB [6] and angular resolution of degrees combine with the narrow 1-nsec pulse width result in accurate target tracking in range and angle.

2.2.2 Basic Target Tracking Algorithms. A range-tracking system makes estimations of future target position and velocity to maintain track. Figure 2.1 shows a block diagram of a basic target tracking algorithm, similar to one seen in the Lab-Volt radar system. The first step includes interrogating the environment to determine if a target exists and comparing collected returns to given threshold settings within the radar. During the gating step, variations of target location are calculated to decide if an observation is part of a previous track or belongs to a new track profile [10]. Gating acts as a coarse classification method to say that the target return is either a candidate for updating the track profile or is the initial observation for a new track profile. The second part of the correlation function makes the final assessment on received returns. During this state, multiple returns falling within the same gate are deconflicted through using either a "nearest-neighbor" or "all-neighbor" approach. In

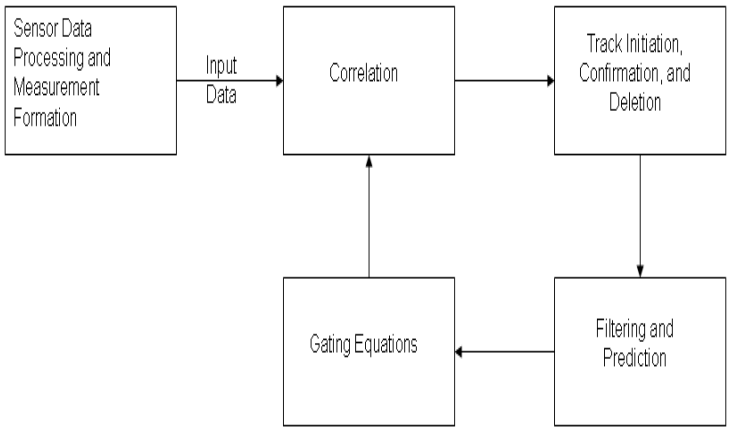


Figure 2.1: Basic elements of a simple tracker system [10]

the "nearest-neighbor" approach, the associated difference, or error value, for each return to a specific range gate is evaluated and the minimal error return is associated with that track, rejecting all other return values [10]. In the "all-neighbor" approach, a weighted sum is applied to all gated returns based on the probability that each return belongs in the given gate [10]. Any observations that are not connected to existing tracks from the previous stage are used for generating new tracking profiles. During this stage in Figure 2.1, confirmation logic is applied to determine if the new tracks are legitimate or should be disregarded. Gate sizing and time period for confirming track are developed to assist the correlation logic in processing track observations. Any track profiles that have not been updated at this point are removed from the tracking scenario. Finally, the remaining target track parameters are updated and future parameter estimates are made regarding the location of the next observation. Prediction estimates are developed through Kalman filtering and covariance matrices are associated to each track profile.

In developing and maintaining tracking profiles, range tracking radars may use detection schemes like Maximum Likelihood Estimators (MLEs) to determine where the target return should come from given the prior knowledge. During this estimation, the assumption given is that the signal is a stationary, bandlimited process,

where the time delay will be fixed for each interval [11]. These MLEs use information on probability of detection (P_d), probability of false-alarm (P_{fa}), error measurement characteristics for the radar, and the target resolution prediction to determine a track profile for a possible detected target. From these tracking parameters, the combinational likelihood for each possible track is found through the binomial distribution of a given track occurring at the current range. Through calculating the target distribution within range cells, the actual return can be compared to either adjust the track profile estimate or adjust the future track distribution space to a smaller region [7,11]. Through numerous MLE iterations, track profiles can be refined by the time increment or approximations in range to develop good track profiles. This methodology can be applied to range tracking techniques, angle tracking techniques or the combination of these and other target parameters.

2.3 Genetic Algorithms

GAs trace back to papers by Holland written in the 1960's which discuss systems that could learn, interact, and adapt to their existing environment [12]. These adaptive systems were designed to take advantage of the biological concepts of natural selection and constant species reproduction. Through competition and innovation in a system, it was noted that an evolutionary algorithm could be developed in an artificial environment to mimic nature's tendencies to find the optimum solution. Genetic algorithms separate themselves from other methods of evolutionary computation based upon three distinct principles:

1. Data representation as bit-strings referred to as chromosomes,
2. The chromosome selection method used is proportional to the population size,
3. The primary method for reproduction with data variation, or creating new data sets, is through crossover between population members.

Understanding the search environment and characterizing good population members through a fitness function exploits the genetic algorithm's distinguishing properties

in finding optimum parameters. The fitness function used by the genetic algorithm operates in the same manner as nature selects the fittest of the species. The genetic algorithm constructs must be understood to appreciate Holland's original ideas of the modern genetic algorithm theory.

2.3.1 Basic Terminology. Genes are the biological building blocks that are represented in this algorithm as data bit-strings. Depending on the data set used, the data points being modeled for optimization can be gray coded for binary strings, represented as permutation matrices, or real-valued like data signals. The chromosome is defined then as an array of parameter values desired to be optimized. If there are N_{par} parameters for the N-dimensional optimization, each given as $p_1, p_2, \cdots, p_N$, the chromosome u_c can be defined as [13]

$$u_c = [p_1 p_2 p_3 \cdots p_N]. \tag{2.4}$$

The parameters can be defined as either continuous or discrete, which can lead to further constraints. If these parameters are continuous, the limits usually represent physical properties of the landscape which bound the problem [13]. Both the chromosome format and the defined constraints dictate the operators used for crossover and mutation, which will be discussed later [14]. To assess a chromosome, there must be a way to determine its performance against the solution set. This performance objective is referred to as a fitness function, which can either be minimized or maximized determined upon how the GA will be used. Chromosome mutation is the random replacement of one allele, or portion of the bit-stream, with another value. These changes are made in small rates as to not change the algorithm's convergence scheme. Contrasting mutation, selection is seen as choosing parent chromosomes to mate to produce offspring, or new possible solution sets. Selection can come in a few different forms, each having unique aspects to development. The first is through a random probabilistic nature, where a percentage of the gathered parents are chosen without preference to their fitness function [14]. Consequently, a tournament-style selection

process would first rank the parent chromosomes based on their fitness values and select a certain percentage of chromosomes for crossover. Crossover itself then operates on two of the selected parents to do either one or two-point crossover, where those points are where data from one parent or the other are spliced together. Depending on the crossover method, these values can result in two children if desired, or just a single child made from the crossed over pieces.

2.3.2 Algorithm Mechanics. Figure 2.2 shows the standard GA functional diagram for optimizing a desired parametric output [15]. Initially, the first parent population is created through either random generation, a heuristic parameter selection or other desired means [12]. During this initial population generation process, the population size must carefully be considered based upon desired computational complexity and prevention of premature convergence. Large population sizes conduct thorough sample-space exploration, but take longer for desired end state converge. In contrast, a small population size does a coarse search through the landscape, but tends to prematurely converge on local maxima/minima instead of finding the global maxima/minima. Once the initial population is generated, the first selection can

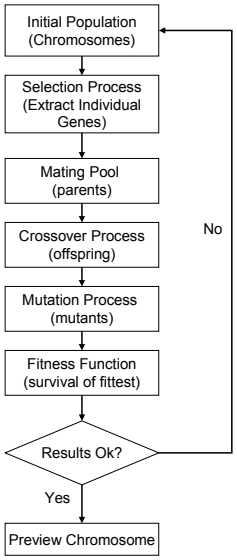


Figure 2.2: Genetic algorithm flowchart

come from any of the selection methods discussed in the previous section. Selection of individuals for the intermediate population come from each chromosome evaluated through the fitness function selected for optimization. Selection of individuals generates the mating pool from those deterministically chosen in the previous step. A probabilistic process determines the number of crossover points in mating along with the mating pairs. As with nature, the possibility exists that the offspring mirror the parents although this event may not be desired. Prevention of exact replication between generations falls in careful crossover probability definition. Gene mutation within the offspring also occurs in a probabilistic manner, which aids in preventing premature convergence. Mutation harms the solution method by changing random genes which could be vital in finding the optimized parameter [12]. Random mutations are used to alter only a small portion of the population. Typically for electromagnetic problems, the reported mutation rate should be on the order of $0.1 - 1\%$ of all genes be mutated for sufficient search results [13, 16].

The resultant offspring are then evaluated by the fitness function used to define the problem space. Fitness function evaluation defines how well that offspring satisfies the condition or possibly multiple conditions desired for optimization. After assigning fitness values to each offspring, the offspring and parents are collected as the current generation in determining solution conditions have been met. Different GAs use different exit criteria based on average fitness performance, best performance achieved or other desired effects. Some GAs operate on a tolerance level, where most chromosomes fall within a given error level of each other, giving the resulting solution. Other GAs operate on finding the singular value within an expected range of values. If the exit criteria is achieved, the optimized chromosome represents the desired solution. Otherwise, the current generation returns back to the selection step shown in Figure 2.2. While previous discussion covered only a single parameter, the multi-objective genetic algorithm transforms this process to handle multi-dimensional problems, like the RGPO optimization [17]. Fitness functions are carefully defined and evaluated in the multi-dimensional case to prevent finding one local extrema that

does not represent the desired global extrema. Careful balance between crossover and mutation prevents the GA from selecting a chromosome with a highly individually fit gene with average overall fitness over chromosomes with near-optimal performance that have a singular bad gene.

The ultimate reason why genetic algorithms are chosen for optimization is for their efficiency involving a large number of parameters. Haupt's paper discusses a couple of common electromagnetic problems that show the difference in operation time. The first problem discussed is minimizing the backscattering-sidelobe level from a grid of perfectly conducting strips. The $2M$ sized array was coded in bit-strings to represent either a strip being present (1) or removed (0) from the grid. Implementing a fully-populated grid of $2M = 40$ strips of width 0.037λ and spacing between strips of 0.1λ, the GA was implemented using 80 chromosomes in the initial population. After eight generations, the GA resulted in a 20-bit gene [13]. An exhaustive search would take 2^{20} possible iterations to check all answers, showing that the GA gave the optimum solution in a much faster manner. His paper continues to explore electromagnetic applications in optimizing sidelobe levels on a nonuniformly spaced array. This time, instead of setting a defined spacing, the chromosome was defined to represent $2N = 48$ elements in an array and each gene was a 3-bit number representing the spacing between elements [13]. Again, instead of implementing 2^{72} possible combinations for an exhaustive search, the GA gives a population-size as a proportion of the number of parameters being optimized on. Papers discussing the application of GAs to electromagnetics give these recommendations for algorithm convergence [13, 16]:

1. Try to use a population size of 10 times the number of bits in an individual chromosome (if bit representation is known). If a significant number of bits do represent a chromosome, use fewer chromosomes in the initial population due to computer RAM issues.

2. Population sizes of 30-100 allow for enough genetic diversity to enable fast convergence while being small enough to execute complicated fitness functions in a timely manner [16].

3. Keep the crossover probability around 0.6-0.9, with most problems resolving successfully with a probability of 0.7 [16].

4. If the algorithm is having problems converging try to "1) increase the number of mutations, 2) increase the number of chromosomes, or 3) add some constraints that you know about from the physics of the problem" [13].

These GA implementation suggestions give a good baseline for operating the MATLAB® GA toolbox with the RGPO and VGPO waveforms. These two articles explore how the GA optimizes specific electromagnetics problems that are relevant to this research. Antenna design and desired sidelobe patterns are significant components of the ECM environment. As discussed earlier, these two components define the radiated signals from and the returned signals to the radar receiver. The ability to optimize specific pieces of the ECM problem suggests that optimization of the waveform dependent upon those components is also possible. Further research gave other current examples of where GA optimization proved useful in resolving similar problems to the methodology given in this thesis. The next section explores literature discussing GA implementations that are similar in nature to this research topic.

2.4 Similar Implementations

While genetic algorithms have been around since the 1960's, their use with radar systems has been limited. Optimization techniques have been explored within the radar environment in various different methods. Numerous optimization techniques have been used in both radar hardware and software development. While some radar optimization problems have been outside the GA realm, their scope still stays in the larger category of evolutionary algorithms, which obey certain properties similar to genetic algorithms, such as they:

1. are a method of optimization or learning,
2. are stochastic in nature, exploring the environment in a non-random search method
3. use "survival of the fittest" analogy of exploring the environment
4. are not fast in computational time, but can scale nicely while being robust about searching,
5. are based upon developing population sets and quality evaluation [17].

Three specific areas explored support the proposed methodology for using the GA optimization technique: Antenna design optimization, radar tracking algorithm optimization, and software designed ECM waveform optimization.

2.4.1 Genetic Algorithms and Antenna Design. Genetic algorithms and other optimization techniques have been used recently to develop better phased array antennas. The paper by Giovanni Golino in 2005 [18] explored using genetic algorithms for defining the optimum antenna division within a phased array antenna for use in producing increased electronic counter-counter measure (ECCM) capabilities. Through exploring the conflicting functionalities between P_d and the Cramer-Rao Lower Bound (CRLB) of the target's estimated angular coordinates, the genetic algorithm was used to find optimum tradeoffs between the two competing values. Using a modified GA from discussion in Section 2.3.2, Golino explores antenna, target, and disturbance characteristics, along with the genetic parameters and objective functions necessary in MATLAB®. Golino states that the experiment used a 64-element array with isotropic radiators divided into 4 separate sub-arrays. The Lab-Volt™ radar contains a phased-array antenna similar to the one used in Golino's research that can validate these experimental results. Evolutionary algorithms such as simulated annealing and genetic algorithms become viable methods of electronic countermeasure development through replicating Golino's research. Golino's research explores using five separate objective functions that cover:

1. the optimum spacing between antenna elements for the (θ_i, ϕ_i) plane,
2. the CRLB estimate of the target's azimuth coordinate in the presence of a main-beam jammer (MBJ),
3. the CRLB estimate of the target's elevation coordinate in the presence of a MBJ,
4. the Peak to Sidelobe Ratio (PSLR) of the developed beam with a MBJ present at $\theta_i = 0$, and
5. the PSLR of the developed beam with a MBJ at $\phi_i = 0$ [18].

From using these five objective functions on the 64-element phased array antenna, 86% of the resulting structures produce the approximations desired. From these structures, human knowledge was required to select the best structure due to physical design limitations that could not be incorporated within the fitness functions given to the genetic algorithm. This article shows that while the genetic algorithm gave desired results in optimizing the five objective/fitness functions, human involvement will still be required in designing of ECM waveforms. Human experience becomes important to ensure that the optimum result is realizable and not something only possible in mathematics [18].

2.4.2 Tracking Radar Development using Optimization Techniques. Tracking radar development recently has relied on using optimization techniques to aid in tuning tracking filters. One such article used the Simulated Annealing optimization technique for automatically tuning tracking filters within a radar receiver [19]. The author motivates the reason behind this research topic: "The task of tuning a radar tracking filter usually involves performing numerous Monte Carlo simulations on a set of 'design-to' scenario trajectories" [19]. Currently, the method for establishing filtering parameters for radar target tracking is done through numerous iterations that can consume valuable financial and manpower resources. Kajenski's use of sim-

ulated annealing to automate this process suggests that this process can be used in the corollary environment, ECM technique generation.

Kajenski develops the framework for the simulated annealing optimization through describing the radar model in detail and then loosely discusses the tracking problem. The author cites a paper from the *1994 American Control Conference* in Baltimore, but never discusses the premise of the control problem and any background or results from this study, which Blackman discusses in the introduction of his book [10]. This benchmark refers to the radar tracking problem, which in Kajenski's article uses a phased-array radar. Experimental results from Kajenski's article discuss the target types emulated in the simulated annealing optimization of the target tracker. The author thoroughly explains how the fitness function used computes the performance score of each target track. These experimental results support the use of optimization techniques in the ECM arena, discussing reasonable search limits, coordinate spaces and the associated population sizes used for problem development [19]. This article furthers the idea for using evolutionary computational methods through its discussion of the time taken to determine tracking filter parameters versus man-hours used in trying to determining a more precise tracking model. This conference paper discusses a method for representing physical limitations like velocity changes and acceleration constants into evolutionary algorithms for optimizing results.

Further exploration of optimization techniques with tracking radars falls in the multiple target track domain. Exploration of multiple target tracks requires developing appropriate coding schemes to account for each target [20]. Similiar to Kajenski's article, the research considers the population size in exploring the landscape. Further development is taken in explaining the fitness function and the P_d based upon multiple possible targets. The research also uses a fitness standardization function to relate current generation fitness to overall fitness. This method applies a linear translation of fitness scores to weight the results based upon progression through the search space [20]. Results from this research suggest the complexity of the genetic algorithm can be simplified of through ordering the fitness function results after applying the

linear regression. Finally, the research implicates that this method can be faster but requires exploration into real-time target tracking systems. This line of thinking supports developing hardware simulation of radar systems using evolutionary algorithms like GA to understanding how to apply physical limitations on the environment.

2.4.3 Previous ECM/GA Work. After understanding how GAs can assist aperture development and create precision tracking filters, the final step to understanding this problem comes in looking at current research done with ECM techniques. Previous work done by Dr. Gary Lamont and his student, Nathan Landis, explored the feasibility of using the GA toolbox located in MATLAB® to exploit ECM techniques [15]. Their experimentation explored using the three input parameters (power factor, ramp length, and ramp peak) for the RGPO ECM technique fitness function. This project explored the effect of mutation and crossover rates on the chromosome population and determining the resolution time needed for finding the best solution. This project helped define the search space needed to solve this ECM optimization problem and defining some of the landscape characteristics. Described below, each input parameter plays a critical role in deceiving the threat radar:

1. Power factor describes the slope of the ramp length to the ramp peak. Larger power factors indicate a quadratic or greater roll-off function.
2. Ramp length describes the roll-off time necessary to get the tracking radar to break lock. The shorter the roll-off time, the quicker the target is trying to escape the radar.
3. Ramp peak describes the maximum delay given to the false return. The larger the ramp peak value, the farther distance the RGPO signal is moved away from the true target return. [15]

Where the issues lie is in understanding the tradeoffs between each parameter. If the power factor becomes too large, the tracking radar could determine that a jammer is present and neglect the RGPO waveform. Depending on the aircraft, a short roll-

off time could signify faster speeds than possible and lead the radar to determine a jammer is present. Finally, if the peak return is too large, the radar determines that the target return could not be that large and starts applying ECCM techniques to remove the jammer.

Testing for this project was limited to software only. Through the use of MATLAB® radar models and the GA toolbox, the computational time necessary to find the optimum solution took between 600 and 1800 seconds [15]. This result shows that the algorithm needs improving to produce results in under 30 minutes. Power factors and ramp lengths take on a large range of values that do not directly correlate to actual radar parameters. The results given state that the desired break-lock times are under 10 seconds, which constitutes $\frac{1}{12}th$ of the engagement time. Better break-lock times are desired for simulation of a real-time environment. Where the results do assist current project development is in exploration of the genetic algorithm itself. One important result from this research is that the population size must be considered carefully in defining the problem. As population sizes increase, the time to resolve an optimum solution increases as well. Carefully bounding the landscape becomes another challenge to explore in this research. Through multiple runs of the genetic algorithm, this ECM/GA effort found that negative fitness values skewed the experimental results, leading to erroneous data. In designing not only the power level, ramp height, and ramp length ranges, careful consideration must be given to physical limitations of the targeted aircraft and the operational environment to ensure that the optimization routine produces desired results. Overall results from this work show that the genetic algorithm can be used in developing ECM techniques against a radar, but must be further extended to hardware-in-the loop simulations, to account for real-world radar parameters.

III. ECM Optimization Problem Methodology

This chapter gives an overview of the methodology used to solve the ECM waveform optimization problem. The first section details how the HILS architecture is developed. This architecture overview serves as a road map for solving the ECM optimization problem. The final section looks at the various components and how they are implemented. These individual components consist of the research efforts covered in Chapters IV and V.

3.1 Architecture Development

The basic HILS architecture for optimization is illustrated in Figure 3.1. In consideration of Figure 1.1, to cause a break-lock event, the HILS architecture must encapsulate the radar operational environment along with recording the desired feedback signals necessary to understand how the tracking radar is reacting to the RGPO signal. Figure 3.1 gives a basic overview of the system for optimization. This architecture depicts the fundamental testing system for developing the optimum ECM waveform. The vector u represents the necessary parameters that are passed to the jammer system. This parameter set is passed forward after the operator selects the desired ECM technique, given as the initialization block. The jammer pod is contained within the radar operational environment block shown in Figure 3.1. The created RGPO signal is then broadcast into the operational environment. The radar system then processes the RGPO signal and the result from the radar signal processor is passed back to the optimization routine to evaluate the results. While the fundamentals of developing this RGPO waveform are seen from this block diagram, this system has problems with handling significant changes. One major problem is that the optimization routine and waveform are embedded within the optimization routine. This implementation serves as a "black-box" method of optimizing the ECM waveform, but does not lend itself to optimization algorithm modifications, ECM waveform, or environmental modifications.

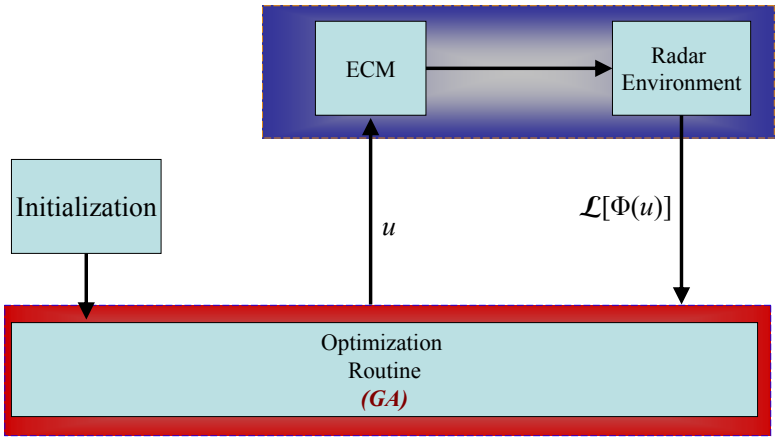


Figure 3.1: ECM technique development basic block diagram.

A better method of developing the ECM optimization architecture is shown in Figure 3.2 [21]. This system separates the optimization routine from the radar operational environment. From initialization, the operator defines the ECM waveform desired for testing. Within the software testing package, shown at the bottom of Figure 3.2, the ECM library then passes the necessary ECM waveform Φ to the jammer in the operational environment and defines the optimization waveform parameters u to the desired optimizer. Furthermore, the ECM library also contains the necessary scoring function $\mathcal{K}$ for the optimization routine to evaluate the outputs from the radar signal processor. This setup allows the optimization routine, which is the genetic algorithm for this research, to operate independently of the ECM technique and radar mode. The genetic algorithm passes the population of chromosomes u for the jammer pod where the jammer waveform $\Phi(u)$ is transmitted into the environment. The radar, contained within the operational environment, processes the ECM waveform, represented as the operator $\mathcal{L}[\Phi(u)]$, and the scoring system acts as the man-in-the-loop observation of the radar response [21]. In other words, the scoring system operates on the radar response as $\mathcal{K}[\mathcal{L}[\Phi(u)]]$. The scoring method $\mathcal{K}$ serves as a method of ordering each member u_c for selection and crossover, as discussed in Section 2.3.2. This architecture has three major advantages:

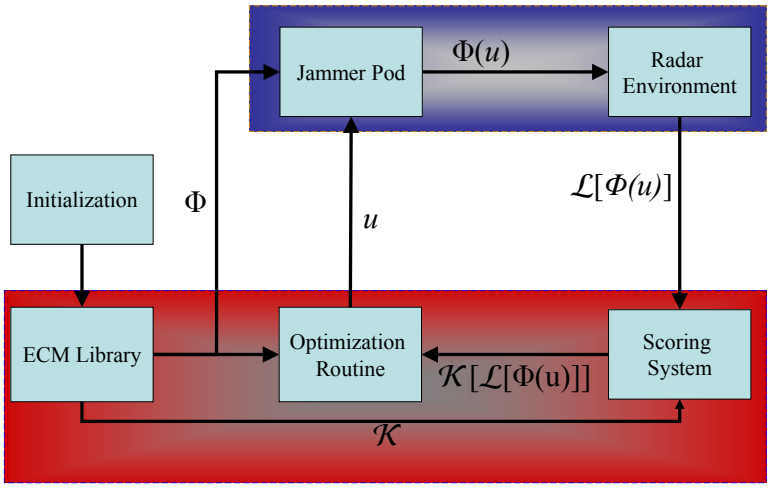


Figure 3.2: ECM optimization architecture block diagram.

1. The operational environment and optimization tools become separable functions. This separability allows the ECM designer to interchange different parts of the system to develop ECM waveform techniques. The software model of the radar environment can be exchanged for physical hardware if it exists. Furthermore, the optimization routine can be exchanged for other techniques, such as Simulated Annealing or Direct Search, depending on the information known about the landscape.

2. A correctly defined scoring function would represent how effective the ECM waveform $\Phi(u)$ was in deceiving the radar tracker. By normalizing the scoring method $\mathcal{K}$ by the maximum pull-off rate within the search space, this value becomes a value with domain [0,1), where 1 represents the ideal case of instantaneous break lock condition. Subsequently, if the optimum parameters results in a value close to 0, that particular ECM waveform model may not be successful for deceiving that radar.

3. Proper waveform definition and optimization bounds represent the physical jammer limitations within the radar environment. These jammer limitations become specific to a particular jammer model or the waveforms implemented by the jammer. Depending on the ECM technique(s) selected, the optimization

routine then becomes a search for the best waveform (RGPO, VGPO, barrage noise, etc.) to use against the particular radar system.

These advantages can be explored further in subsequent research. The next section discusses the individual component implementation and how the research handled each piece.

3.2 Architecture Components

This section examines the various blocks of Figure 3.2 and how they were developed in this research. Each subsection describes the methods used to develop these systems and how they were integrated into the entire simulation.

3.2.1 ECM Library. The electronic countermeasure library contains the various mathematical equations used to represent the waveform. An example mathematical equation stored in the ECM library is the linear acceleration model, detailed explicitly in section 4.2.1, as:

$$r(t) = \frac{(R_{\max} - R_o)}{2{T_w}^3}t^3 - \frac{(R_{\max} - R_o)}{{T_w}^2}t^2 + \frac{3(R_{\max} - R_o)}{2T_w}t + R_o \tag{3.1}$$

The parameter set u for implementing equation (3.1 is expressed as:

$$u = \left[\begin{array}{cccccccc} R_{\min} & R_{\max} & T_w & f & \dot{R}_{\max} & \dot{R}_{\min} & A_o & A_{\min} \end{array} \right], \tag{3.2}$$

where each parameter contained in u represents a physical property emulated by the deceptive waveform. Table 3.1 lists each with a brief description. The parameter set listed in Table 3.1, which is explained in-depth in section 4.2.1, serves as an example of u passed to the GA in the optimization block. A similar parameter set exists for the VGPO waveform, explained in section 4.3. The library serves as a catalog of all associated $\Phi(u)$ waveforms that exist for the modeled jammer, such as the RGPO and VGPO waveform expressions, and their scoring methods comprise the ECM library.

Table 3.1: RGPO Parameter Set Definitions

Variable	Definition
$R_{\max}$	Maximum pull-off range
$R_{\min}$	Initial pull-off range
T_w	RGPO profile walk-off time
f	RGPO profile number
$\dot{R}_{\max}$	Maximum pull-off range rate
$\dot{R}_{\min}$	Minimum pull-off range rate
A_o	Initial jammer signal amplitude
$A_{\min}$	Minimum jammer signal amplitude

The next block covered, the optimization routine, explains the optimization routine used for simulation.

3.2.2 Optimization Technique. This section explains the optimization technique block in Figure 3.2. The reviewed literature confirms that the genetic algorithm can adequately optimize the electronic countermeasure waveforms developed for this architecture. Section 5.1 explains the MATLAB® GA toolbox which is used to implement the optimization routine. This toolbox was selected for the following reasons:

1. The radar asset modeled for this research was developed in Simulink. MATLAB® and Simulink work together and provide seamless integration of the jammer model, genetic algorithm toolbox, and the radar model for proof of concept.
2. MATLAB® is hosted on the Tektronix Arbitrary Waveform Generator, Real-Time Spectrum Analyzer and the Digital Signal Oscilloscope contained in the Radar Analysis Laboratory (RAIL). GA, or other optimization methods in MATLAB® allows later research to replace these simulations with hardware components with minimal effort.
3. Significant documenting on the Genetic Algorithm toolbox reduces the learning curve to prove ECM optimization. The Genetic Algorithm toolbox also contains other search methods for further research of the optimization-waveform pair.

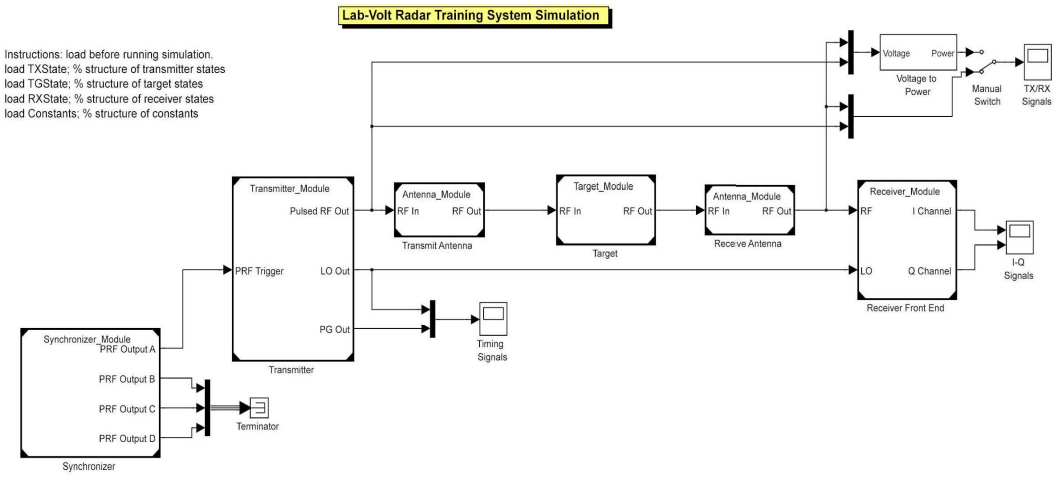


Figure 3.3: Simulink block diagram of Lab-Volt™ radar [22].

The GA toolbox allows for user-defined fitness functions, which can be developed in other MATLAB® code, Simulink models, or hardware development.

3.2.3 Jammer Pod. The MATLAB® modeled jammer pod was developed from the Lab-Volt™ Jammer specifications [23]. This code, discussed in Section 4.2.4, uses the actual mathematical expressions from the ECM Library function and implements the transmitted signal against a specified radar platform. This implementation allows the operator to specify the jammer pod's operational mode from outside the operational environment. Furthermore, the jammer pod parameters are easily changed for mimicking a specific asset. Use of the Lab-Volt™ system allows for validation of the MATLAB® code from which further research could then develop more sophisticated models.

3.2.4 Radar Operational Environment. Figure 3.3 shows the Simulink block diagram of the Lab-Volt™ radar system. This Simulink development is based on research by 2nd Lt. Oscar Mayhew, who characterized various components of the Lab-Volt™ system [6]. The work in [6] validates the Simulink model developed by Maj. Michael Saville. This radar model contains the basic radar operational components of: Radar Transmitter, Radar Receiver, Synchronizer and Antenna Gain. As

seen in Figure 3.3, the target block models the true target position and cross-section in the environment. The MATLAB® jammer model is inserted after this block, prior to reception by the receiving antenna. Once the signal is received at the antenna block, the true and false target information exist as found in the operational environment. $\mathcal{L}[\Phi(u)]$ represents the signal transmitted into the radar environment, the environment effects imposed prior to reception, and the radar signal processer manipulating the data into a video output. Although these operations in the radar environment are nonlinear functions, Cheney's paper states that the appropriate approximations lead to a linearized problem [24]. Cheney's discussion on strip-mode SAR starts with the Maxwell's equation representation. Using the Born approximation for the scattering solution, the matched filter processing representation can be given as a linear equation. The result from this paper allows for $\mathcal{L}[\Phi(u)]$ to be a linear operator and be optimized using techniques such as the GA. After the signal is passed through the Dual-Channel Sampler, the digital scope, represented as the returned Simulink output to MATLAB® captures the data for scoring function evaluation.

3.2.5 Scoring Function. The scoring function operates on the collected data from the radar signal processor and makes a decision on whether the false target 'spoofed' the radar system. Once the signals have passed through the Dual-Channel Sampler, the received signals are integrated over the collection time in step sizes of the radar's pulse width to determine which range bin the target returns fall into. This radar signal processing output $\mathcal{L}[\Phi(u)]$ is then evaluated to determine how successful the ECM waveform was against the radar system. Figure 3.4 shows an example of the radar scoring function. The matrix on the right shown in Figure 3.4 represents the collected data from the output of the dual-channel sampler. The scoring function waits for the jammer signal created by u_c to propagate through the environment for the entire walk-off time before producing a result. The range of $m = [1, M_{\text{J}}]$, with M_{J} representing the number of CPIs collected during the simulation. At a minimum, M_{J} must be larger than the number of pulse repetition intervals (PRIs) required for

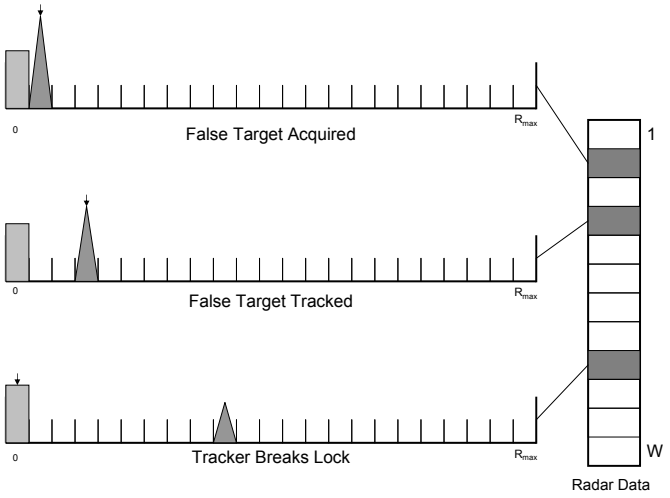


Figure 3.4: GA scoring function example.

the scoring function to appropriately evaluate each u_c. Three data slices from the Dual-Channel Sampler are shown on the left of Figure 3.4. Each row of the collected data matrix from the Dual-Channel Sampler has been truncated to contain only the walk-off range for the ECM signal. The individual rows span from t=[0:T_w], where time "zero" represents the range bin where the true target exists. As described in section 2.2.2, determination of which target the radar is tracking depends on the range cell where the most average power exists. The top row shown in Figure 3.4 shows the radar tracking the false target cell on initialization of the ECM waveform. The center row shown in the figure shows that over time, the target is moving with decreased power returned to the target but still holds the target tracker. When the target tracker finally breaks lock, depicted in the bottom row of Figure 3.4, the false target cell no longer contains enough power to deceive the radar receiver, and the victim radar has lost track of the real target. The scoring function $\mathcal{K}$ then records the row where target track was broken. This value $\mathcal{K}[\mathcal{L}[\Phi(u)]]$ is normalized by M_{J}, to give an effective ratio for that chromosome u_c. Once all chromosomes are evaluated by the scoring function, the values are returned back to the optimization routine to determine future generations if the optimum solution has not been found.

3.3 Methodology Summary

This research effort establishes the validity of the ECM techniques generation shown in Section 3.1. The HILS architecture shown in Figure 3.2 separates the optimization technique, developed in this research using the MATLAB® GA toolbox, from the radar development environment, established in the Simulink block diagram in Figure 3.3. Architecture development in this manner promotes modular design, allowing for plug and play capabilities of different radar models, jammer pods, ECM library functions and optimization methods as the operator desires. While the system architecture is developed with a known radar system, this system can also be implemented with minimum knowledge of the victim radar. The ECM library developed for this architecture exists under the general mathematical cases for RGPO and VGPO, lending itself for use for any system. Furthermore, if the optimization parameter landscape for the victim radar lends itself to search methods with known shapes, the genetic algorithm block can be replaced for a deterministic search method like a direct search or a terrain climbing approach like simulated annealing. The next chapter looks at the development of basic library functions for the ECM jammer pod and the associated parameter set that is passed to the optimization routine. Chapter V evaluates the ECM waveforms developed for the RGPO signal.

IV. Electronic Countermeasure Waveform Modeling

The first section in this chapter discusses the foundations of electronic countermeasure technique development. The second section in this chapter discusses the mathematical model and MATLAB® implementation of the generalized Range Gate Pull-Off technique. The last section in this chapter discusses the mathematical derivation of Velocity Gate Pull-Off and the subsequent MATLAB® implementation using the space-time adaptive processing paradigm.

4.1 ECM Development Theory

This section takes a closer look at the typical EW scenario, discussed earlier in Section (1.1). In the engagement scenario, shown in Figure 1.1, the desired ECM waveform deceives the victim radar by introducing additional radar signals that have characteristics of additional targets with different locations and velocities. Once the radar tracker breaks lock from the true target return, the radar must reacquire the target via its search mode. By causing the threat radar to initiate a search mode, the targeted aircraft buys important time to remove itself from the engagement or has the ability to reverse the engagement situation. Judicious use of the ECM waveforms enables the targeted platform to change the scenario.

One specific type of jammer that seeks to achieve this goal is the deception/repeater jammer, which masks the real target by injecting suitable modified replicas of the real signal into the victim system [25]. As previously mentioned, the deception jammer does not flood the EM spectrum with external noise but provides false ranging or velocity information to the victim system. Figure 4.1 shows the system architecture of a basic repeater jammer, seeking to replicate the desired capture signal. The RF signals intercepted at the receiving antenna are first amplified to allow the receiver circuit to make the appropriate decision on the received signal. Decisions are made through the control circuit regarding what type of ECM response is desired to protect the host asset. Upon determination of the ECM waveform necessary, the amplitude and phase modulation block injects the appropriate modulation(s) neces-

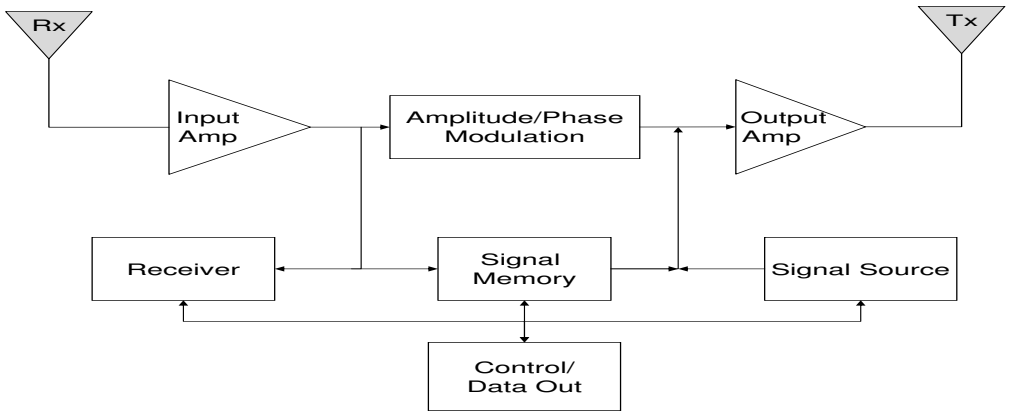


Figure 4.1: Repeater jammer block diagram [26].

sary to replicate the received waveform [26]. The output amplifier boosts the signal prior to transmit. The signal memory and signal source circuits are disabled by the control source in the repeater case. Neither circuit is necessary because the input signal is being regenerated with appropriate amplitude/phase shifts and retransmitted, without prior history needed. Specific types of deception jamming require these circuits, which will be discussed in later sections. Within this system, the transmitter output is directly proportional to the received signal. This system, known as a constant gain system, is designed such that the transmitter power output is proportional to the received signal power level, preventing any feedback to the input receiver and subsequent unnecessary oscillation within the circuitry.

An extension of the repeater jammer is to operate the same circuit shown in Figure 4.1 in a different configuration for deception signals. Instead of passing the signal through the amplitude/phase modulation block, the circuit uses the signal memory block to determine coherence. This configuration, known as a transponder system with constant gain, uses time delayed copies of the original to rebroadcast back into the environment. Coherence becomes important as it relates "... the accuracy with which the intercepted signal can be reproduced in its carrier frequency, which includes any frequency or phase modulation contained with the intercepted signal" [26]. Deception techniques, discussed below, further stress the significance of

coherence in EW signals. The signal memory and receiver circuits help regenerate the power level necessary such that the maximum power out of the transmitter mirrors the intercepted signal level. Transponders prevent feedback signals from ruining the intercepted signal as the transmit signal is gated from the receiver circuit during transmission.

Through different variations of Figure 4.1, the desired operational mode of a deception jammer provides realistic false target information to deceive the victim radar. Similarity between various deception/repeater jammer operational block diagrams allows the same hardware to create various ECM waveforms without additional equipment to a jammer pod. Instead, understanding the necessary signal coherence and the necessary amplitude/phase modulations for waveform generation allows for an optimization routine to select the best waveform for a specified engagement. The following sections will describe the mathematical formulation for both RGPO and VGPO waveforms, which subsequently lends itself to MATLAB® modeling.

4.2 Generalized RGPO Techniques

One specific deception waveform found to be effective against pulse-Doppler tracking radars is the range-gate pull-off (RGPO) waveform [27]. This section explores the generalized RGPO modeling technique to be optimized by a genetic algorithm. First, a mathematical discussion is presented on how false targets are generated by the jammer and the associated range delays are induced. In generating the associated range delays, the amplitude scaling must also be considered to ensure proper false target representation. Next, the physical limitations of the mathematics are discussed to shed light on the boundaries that must be considered by the optimization method. Following, is an explanation of how MATLAB® can implement the RGPO jammer model. Finally, the MATLAB® mathematical model is compared to the Lab-Volt™ jammer pod accompanying the radar training system.

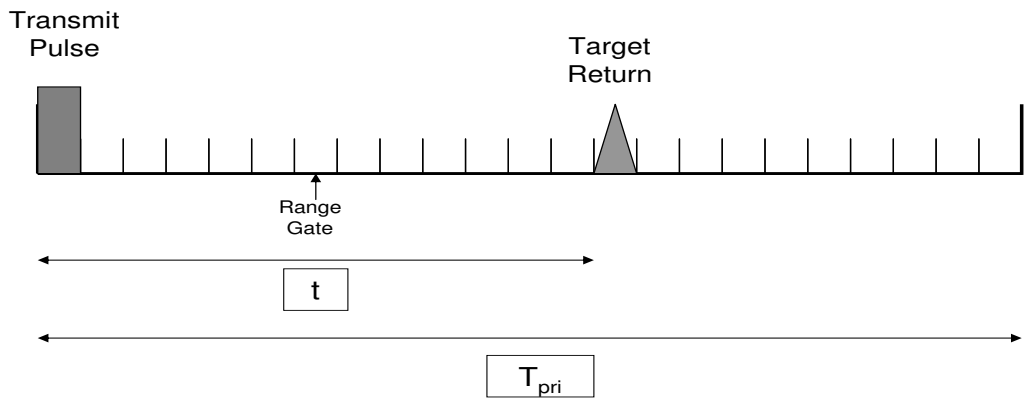


Figure 4.2: Radar range measurements.

4.2.1 Mathematical Background. A better understanding of how the RGPO jammer pulls the tracker off the true target comes from looking at how the tracking radar operates. Figure 4.2 shows how the range bins are arranged within the PRI. In the tracker's PRI (T_{pri}), the pulse waveform is transmitted towards and reflected from a target within the space volume surveyed by the radar [26]. The range (R) at which the target is detected at is a function of the time (t) it takes the pulse to travel to and from the radar, and is:

$$R = \frac{ct}{2}, \tag{4.1}$$

where c represents the speed of light. The radar's reference point for when time begins in the PRI is determined to be at the leading edge of the pulse shown in Figure 4.2. The PRI can then be segmented into a series of range bins, arranged contiguously after the pulse would be transmitted by the radar [26]. Each range bin is one pulse width τ in length, which represents the radar's range resolution [9]. It can be seen from Figure 4.2 that if targets return from more than one position they will show up in different gates, while multiple returns within τ will be lumped together in the same range bin. The following three subsections cover three different profiles: Constant Velocity, Constant Acceleration, and Linear Acceleration. The following mathematical models were selected for physical limitations of realistic target motion. The technique and development applies to general mathematical forms for ECM.

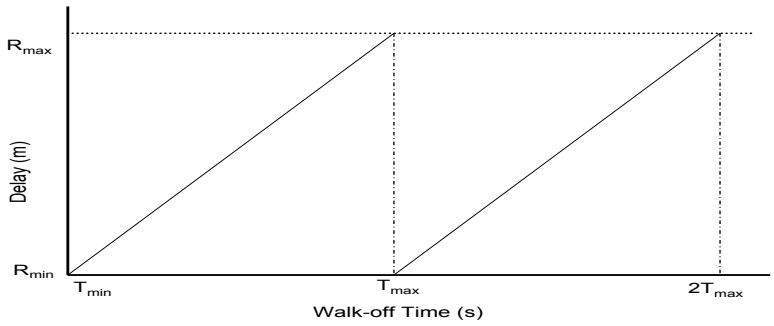


Figure 4.3: Linear RGPO profile, adapted from [26].

Although any mathematical function can be used, the polynomial orders of one, two or three are chosen to agree with known equations of motion. Subsequently, these models uses physical properties to move the false target away a given distance.

4.2.1.1 Constant Velocity Profile. By understanding how the radar processes detected targets, false target range delays can be developed appropriately within the operational environment. From the radar range bin setup illustrated in Figure 4.2, the next step in developing the RGPO profile is looking back at the physical motion being represented in moving range bins. The equations for position $r(t)$, velocity $v(t)$, and acceleration $a(t)$ are defined as [28]:

$$r(t) = \frac{\partial}{\partial t} v(t), \tag{4.2}$$

$$v(t) = \frac{\partial}{\partial t} a(t), \tag{4.3}$$

$$r(t) = \frac{\partial^2}{\partial t^2} a(t). \tag{4.4}$$

Equations (4.2)-(4.4) become the basis for developing the desired false target motion with known profile parameters. Figure 4.3 depicts a linear range profile to move the false target away. The linear RGPO profile shown in Figure 4.3 can be modeled with the following equations:

$$r(t) = C_1 t + C_0, \tag{4.5}$$
$$v(t) = \frac{\partial}{\partial t} r(t) = C_1, \tag{4.6}$$
$$a(t) = \frac{\partial}{\partial t} v(t) = 0, \tag{4.7}$$

which as shown in Figure 4.3 shows a linear walk-off with constant velocity and no acceleration. The constants C_0 and C_1 are solved for by specifying boundary or initial conditions of the RGPO profile. The equation is a first degree polynomial making f equal or proportional to the degree. Figure 4.3 depicts the conditions for the initial delay R_o, maximum delay $R_{\max}$, and the total walk-off time T_w. These initial conditions are then expressed in the following manner:

$$r_(T_{\min}) = R_o, \tag{4.8}$$
$$r(T_{\max}) = R_{\max}, \tag{4.9}$$
$$T_w = T_{\max} - T_{\min}. \tag{4.10}$$

Substitution of Equations (4.8)-(4.9) into (4.5)-(4.6) gives the following conditions:

$$r(T_{\min}) = C_1 T_{\min} + C_0 = R_o, \tag{4.11}$$
$$r(T_{\max}) = C_1 T_{\max} + C_0 = R_{\max}. \tag{4.12}$$

Equations (4.11)-(4.12) further simplify with the understanding that the first pulse should have no time delay from the original pulse. This equates to $T_{\min} = 0$, which applied to (4.11) results in

$$R_o = C_1 \cdot 0 + C_0 = C_0. \tag{4.13}$$

Furthermore, this also makes $T_w = T_{\max}$ because the time offset needs to match the engagement start time for coherence. Replacing the calculated C_1 from (4.13) into

(4.12) gives the other unknown:

$$\begin{aligned} R_{\max} &= C_1 T_{\mathrm{w}} + R_o, && (4.14) \\ R_{\max} - R_o &= C_1 T_{\mathrm{w}}, && (4.15) \\ \frac{R_{\max} - R_o}{T_w} &= C_1. && (4.16) \end{aligned}$$

After calculation of both constants, the linear position model for Figure 4.3 is expressed as:

$$r(t) = \frac{R_{\max} - R_o}{T_w} t + R_o. \tag{4.17}$$

Equation 4.17 represents the continuous time representation of range profile to pull the false target away in a linear fashion. Furthermore, it is understood that the RGPO jammer operates in a discrete manner and can not fully implement this continuous signal. Discretization of this signal comes from looking back at tracking radar, illustrated in Figure 4.2. The time step given by each range bin can be defined as

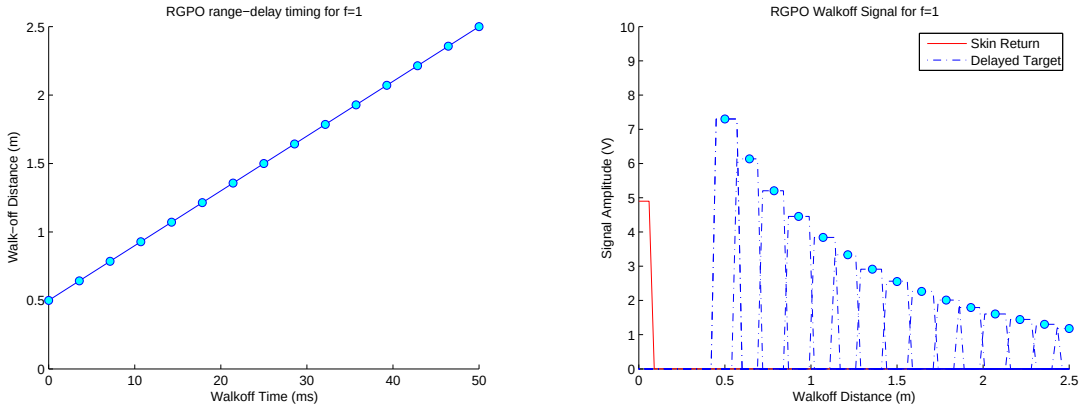
$$t = m T_{\mathrm{PRI}}, \tag{4.18}$$

where T_{PRI} is the radar's PRI. The variable m denotes the specific discrete walk-off step within the total engagement time. Substitution of (4.18) to (4.17) gives the discrete time-step function:

$$r_m = \frac{R_{\max} - R_o}{T_w} (m T_{\mathrm{PRI}}) + R_o, \tag{4.19}$$

with the range of values for $m = [0, 1, \cdots, M_{\mathrm{J}}]$. As discussed briefly in Section 3.2.5, M_{J} represents the number of tracking radar PRI's required to walk the target a distance $R_{\max}$. This number is calculated as the ratio of walk-time to the processing time, given as:

$$M_{\mathrm{J}} = \left\lfloor \frac{T_w}{T_{\mathrm{PRI}}} \right\rfloor, \tag{4.20}$$



(a) RGPO Range-Delay Profile.

(b) RGPO Walk-off Signal.

Figure 4.4: Constant velocity RGPO profile.

where the floor function is used to round down the number of PRIs that fit within the necessary walk-off time. Substitution of (4.17) and (4.20) into (4.17) results in the discrete representation of the range delay signal as

$$r_m = (R_{\max} - R_o)\frac{m}{M_{\mathrm{J}}} + R_o. \tag{4.21}$$

Figure 4.4 shows the resulting constant velocity RGPO signal modeled with equation (4.21) where $T_w = 50$ ms, $R_o = 0.5$ m, and $R_{\max} = 2.5$ m. The range-delay profile shown in Figure 4.4(a) corresponds to a constant velocity of $40\frac{\mathrm{m}}{\mathrm{s}}$. This constant velocity can be seen in the delayed signal shown in Figure 4.17, pulses, marked with circles, are equally spaced and correspond to given distances away from the target shown in Figure 4.4. The relative distance shown in Figure 4.4(b) is from the true target return, shown as a solid line. The signal amplitude decreases in a $\frac{1}{R^2}$ proportion as distance increases, similar to what would be experienced by the true signal return. The amplitude modulation for the RGPO jammer signal is discussed further in Section 4.2.2. While this signal is the most basic of delayed target time-signals, this development becomes the basis for larger, more-complicated signals that compensate for other physical characteristics.

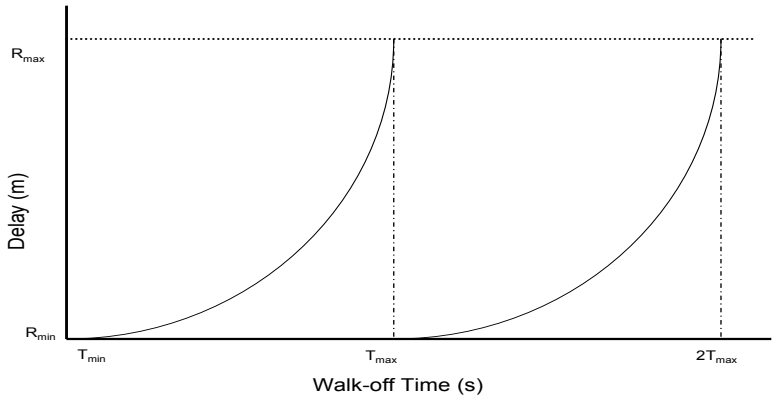


Figure 4.5: Generic constant acceleration RGPO range/delay profile.

4.2.1.2 Constant Acceleration RGPO Signal. The most common RGPO signal to model is the constant acceleration profile, which results in a parabolic curve in the range/delay profile shown in Figure 4.5. This profile represents a constant thrust profile described as:

$$F = ma(t), \tag{4.22}$$

which makes the function $a(t)$ a constant value, C_o. From equation (4.22), the position and velocity equations in (4.2) and (4.3) are rewritten accordingly as:

$$\begin{aligned} a(t) &= C_2, & (4.23)\\ v(t) &= C_2 t + C_1, & (4.24)\\ r(t) &= \frac{C_2 t^2}{2} + C_1 t + C_0. & (4.25) \end{aligned}$$

Equations (4.2)-(4.4) resemble the constant acceleration equations (4.23)-(4.25) but with an added time-dependence to the velocity and position equations. Solving this differential equation requires additional information from the constant velocity model discussed previously. The additional constant added is the maximum range rate $V_{\max}$,

giving the following boundary conditions:

$$V_{\max} = \frac{R_{\max}}{T_w}, \tag{4.26}$$

$$r(T_{\min}) = R_o, \tag{4.27}$$

$$r(T_{\max}) = R_{\max}, \tag{4.28}$$

$$T_w = T_{\max} - T_{\min}. \tag{4.29}$$

From these initial conditions, $T_{\min} = 0$ still holds true for the constant acceleration model:

$$R_o = \frac{C_2 T_{\min}^2}{2} + C_1 T_{\min} + C_0, \tag{4.30}$$

$$= \frac{C_2 \cdot 0^2}{2} + C_1 \cdot 0 + C_0, \tag{4.31}$$

$$= C_0. \tag{4.32}$$

Substitution of (4.29) into (4.24) and (4.25) determines the coefficients C_1 and C_0:

$$v(T_w) = C_2 T_w + C_1, \tag{4.33}$$

$$r(T_w) = \frac{C_2 T_w^2}{2} + C_1 T_w + R_o. \tag{4.34}$$

Understanding that $v(T_w) = \dot{R}_{\max}$, (4.33) can be rewritten in terms of C_1 as

$$V_{\max} = C_2 T_w + C_1, \tag{4.35}$$

$$C_1 = \frac{R_{\max}}{T_w} - C_2 T_w. \tag{4.36}$$

C_2 can be solved by substituting (4.36) into (4.34) for C_1:

$$r(T_w) = \frac{C_2 {T_w}^2}{2} + \left(\frac{R_{\max}}{T_w} - C_2 T_w \right) (T_w) + R_o. \tag{4.37}$$

Equation (4.37) simplifies after substitution of (4.27) and (4.28) and removing like terms in the following manner:

$$R_{\max} = \frac{C_2 {T_w}^2}{2} + R_{\max} - C_2 {T_w}^2 + R_o, \tag{4.38}$$

$$-R_o = -\frac{C_2 {T_w}^2}{2}, \tag{4.39}$$

$$C_2 = \frac{2R_o}{{T_w}^2}. \tag{4.40}$$

C_2 is the constant acceleration modeled for the initial equations (4.23)-(4.25). The final constant, C_1, solved after substituting C_2 back into (4.36), results in:

$$C_1 = \frac{R_{\max}}{T_w} - \left(\frac{2R_o}{{T_w}^2}\right) T_w, \tag{4.41}$$

$$= \frac{R_{\max} - 2R_o}{T_w}. \tag{4.42}$$

Constant C_1 represents the minimum range rate for the false target to move with minimum velocity $V_{\min}$ during the constant acceleration profile. Substitution of the constants back into the original range equation (4.25) gives the constant acceleration model of the RGPO signal:

$$r(t) = \frac{2R_o}{{T_w}^2} t^2 + \frac{R_{\max} - 2R_o}{T_w} + R_o, \tag{4.43}$$

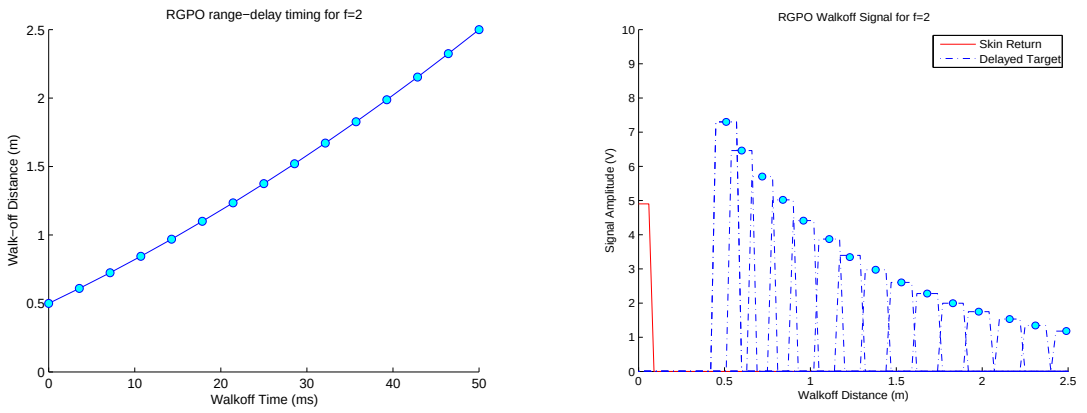
and the discrete case for efficient modeling and simulation:

$$r_m = \frac{2R_o}{{M_{\mathrm{J}}}^2} m^2 + \frac{R_{\max} - 2R_o}{M_{\mathrm{J}}} m + R_o. \tag{4.44}$$

The parameters necessary for modeling the jammer signal are then represented as:

$$\bar{u} = [T_w, R_o, R_{\max}, V_{\max}, V_{\min}]. \tag{4.45}$$

The derivation of $V_{\min}$ and $V_{\max}$ shows their dependence on either R_o or $R_{\max}$. This dependency shows that while the parameters passed to the ECM Jammer Pod are



(a) RGPO Range-Delay Profile.

(b) RGPO Walk-off Signal.

Figure 4.6: Constant acceleration RGPO profile.

contained in $\bar{u}$, the minimum parameters required for implementing the constant acceleration profile are contained in:

$$\bar{u}_{\min} = [T_w, R_o, R_{\max}]. \tag{4.46}$$

Figure 4.6 illustrates this minimum parameter set as shown in (4.46), using the same constant parameters ($R_{\max} = 2.5\text{m}, R_o = 0.5\text{m}, T_w = 50\text{ms}$) as used in the constant velocity model. Figure 4.6(a) shows the range-delay profile developed from (4.44) with a parabolic curve similar to Figure 4.5. This profile shows up upon close examination of the RGPO walk-off signal in Figure 4.6(b). The first few pulses are bunched close together as the range separation between pulses is small. The spacing between pulses in distance grows over the course of the profile, showing the parabolic change. This profile shows a trend in the RGPO signal and its dependence on specific parameters, given in (4.46). The linear acceleration model continues to show this dependence that lends itself towards a generic RGPO profile given the desired f value.

4.2.1.3 Linear Acceleration RGPO Signal. The final RGPO signal represented is the linear acceleration model. The linear profile is expressed as

$$a(t) = C_3 t + C_2. \tag{4.47}$$

Integrating this function over time to achieve the velocity equation and then once more for the position equation results in the following forms:

$$v(t) = \frac{C_3 t^2}{2} + C_2 t + C_1, \tag{4.48}$$

$$r(t) = \frac{C_3 t^3}{6} + \frac{C_2 t^2}{2} + C_1 t + C_0. \tag{4.49}$$

These equations represent the polynomial factor $f = 3$ with 4 unknown constants C_0 to C_3. The addition of another constant C_3 requires an additional initial condition to solve for all unknowns. In this case, maximum acceleration $A_{\max}$ is defined along with all other previously stated initial boundary conditions:

$$\begin{aligned} A_{\max} &= \frac{R_{\max} - R_o}{{T_w}^2}, \\ V_{\max} &= \frac{R_{\max} - R_o}{T_w}, \\ r(T_{\min}) &= R_o, \\ r(T_{\max}) &= R_{\max}, \\ T_w &= T_{\max} - T_{\min}, \\ T_{\min} &= 0. \end{aligned} \tag{4.50}$$

The $f = 3$ case mirrors the constant acceleration ($f = 2$) and constant velocity ($f = 1$), lending to solving for the four unknowns in a similar manner as before. First, $t = 0$ is substituted into (4.49), solving for C_3:

$$r(0) = \frac{C_3 \cdot 0^3}{6} + \frac{C_2 \cdot 0^2}{2} + C_1 \cdot 0 + C_0, \tag{4.51}$$

$$R_o = C_0. \tag{4.52}$$

Next, the acceleration at $t = T_w$ in (4.47), results in

$$A_{\max} = C_3(T_w) + C_2. \tag{4.53}$$

Equation (4.53) is rewritten with $R_{\max}$ and R_o explicitly and expressed in terms of C_2:

$$\frac{R_{\max} - R_o}{{T_w}^2} = C_3(T_w) + C_2, \tag{4.54}$$

$$C_2 = = \frac{R_{\max} - R_o}{{T_w}^2} - C_3(T_w). \tag{4.55}$$

The next step is to substitute $V_{\max}$ into (4.48), subsequently applying the condition $t = T_w$. This substitution results in

$$\frac{R_{\max} - R_o}{T_w} = \frac{C_3(T_w)^2}{2} + C_2(T_w) + C_1, \tag{4.56}$$

which when (4.55) replaces C_1 equates to

$$\frac{R_{\max} - R_o}{T_w} = \frac{C_3{T_w}^2}{2} + \left(\frac{R_{\max} - R_o}{{T_w}^2} - C_3T_w\right)(T_w) + C_1. \tag{4.57}$$

Further simplification of (4.57) by reducing like terms leaves the equation as

$$\frac{R_{\max} - R_o}{T_w} = \frac{C_3{T_w}^2}{2} + \frac{R_{\max} - R_o}{T_w} - C_3{T_w}^2 + C_1 \tag{4.58}$$

$$0 = -\frac{C_3{T_w}^2}{2} + C_1 \tag{4.59}$$

$$C_1 = \frac{C_0{T_w}^2}{2}. \tag{4.60}$$

Using the initial condition $r(T_w) = R_{\max}$ in equation (4.49) results in

$$R_{\max} = \frac{C_3{T_w}^3}{6} + \frac{C_2{T_w}^2}{2} + C_1T_w + R_o. \tag{4.61}$$

With (4.55) and (4.60) written in terms of C_2 and C_1 respectfully, (4.61) can then be expressed as:

$$\begin{aligned} R_{\max} &= \frac{C_3 {T_w}^3}{6} + \left(\frac{R_{\max} - R_o}{{T_w}^2} - C_3 T_w \right) \frac{{T_w}^2}{2} + \cdots \\ &\quad \left(\frac{C_3 {T_w}^2}{2} + C_1 \right) T_w + R_o \end{aligned} \tag{4.62}$$

$$R_{\max} - R_o = \frac{C_3 {T_w}^3}{6} + \frac{R_{\max} - R_o}{2} - \frac{C_3 {T_w}^3}{2} + \frac{C_3 {T_w}^3}{2} \tag{4.63}$$

$$\frac{R_{\max} - R_o}{2} = \frac{C_3 {T_w}^3}{6}. \tag{4.64}$$

One final simplification of (4.64) solves for the value C_3.

$$C_3 = \frac{3(R_{\max} - R_o)}{{T_w}^3} \tag{4.65}$$

Using C_3, C_2 and C_1 are determined. The first value to solve is C_2, by applying C_3 to (4.55).

$$C_2 = \frac{R_{\max} - R_o}{{T_w}^2} - \frac{3(R_{\max} - R_o)}{{T_w}^3} T_w \tag{4.66}$$

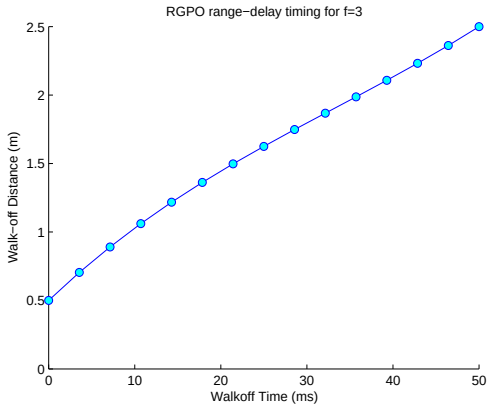
$$= -\frac{2(R_{\max} - R_o)}{{T_w}^2} \tag{4.67}$$

Equation (4.67) represents the minimum acceleration for the linear acceleration profile $A_{\min}$. Although this quantity was not expressed initially in defining the system, $A_{\min}$ could be specified instead. The final quantity to solve, C_1, is determined in the same manner as C_2 after substitution of C_3 into (4.60).

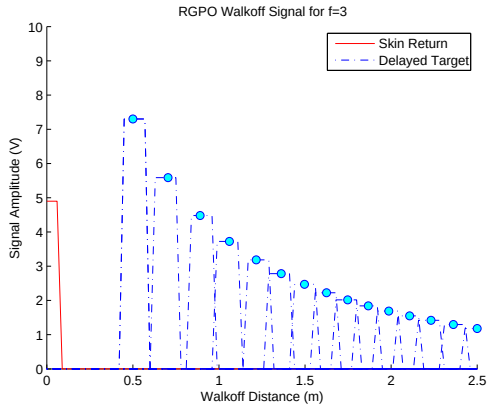
$$C_1 = \left(\frac{3(R_{\max} - R_o)}{{T_w}^3} \right) \frac{{T_w}^2}{2} \tag{4.68}$$

$$= \frac{3(R_{\max} - R_o)}{2T_w} \tag{4.69}$$

C_1 represents the minimum range rate V_{min} for the RGPO profile with linear acceleration. The range delay signal $r(t)$ can then be solved by inserting the constants back



(a) RGPO Range-Delay Profile.



(b) RGPO Walk-off Signal.

Figure 4.7: Linear acceleration RGPO profile.

into (4.49).

$$r(t) = \frac{(R_{\max} - R_o)}{2{T_w}^3}t^3 - \frac{(R_{\max} - R_o)}{{T_w}^2}t^2 + \frac{3(R_{\max} - R_o)}{2T_w}t + R_o \tag{4.70}$$

The expanded parameter set for implementing the linear acceleration RGPO profile $\bar{u}$ includes the following:

$$\bar{u} = [T_w, R_o, R_{\max}, V_{\max}, V_{\min}, A_{\max}, A_{\min}]. \tag{4.71}$$

The derivation of $V_{\min}, V_{\max}, A_{\min}, A_{\max}$ all show a dependence on the initial range parameters of either R_o or $R_{\max}$ or both. This dependency continues the trend that the parameters passed to the ECM Jammer Pod are contained in $\bar{u}$, but again can be simplified to the minimum parameters:

$$\bar{u}_{\min} = [T_w, R_o, R_{\max}]. \tag{4.72}$$

Figure 4.7 shows the linear acceleration model RGPO signal implemented with dependence on $\bar{u}_{\min}$. Mathematical representation for the MATLAB® jammer implementa-

tion comes from the discrete time signal notation substitution as in (4.17)-(4.21).

$$r(w) = \frac{(R_{\max} - R_o)}{2{M_{\text{J}}}^3} m^3 - \frac{(R_{\max} - R_o)}{{M_{\text{J}}}^2} m^2 + \frac{3(R_{\max} - R_o)}{2M_{\text{J}}} m + R_o \tag{4.73}$$

The constant acceleration RGPO range/delay profile shown in Figure 4.7(a) exhibits the cubic function as expected from the general solution in (4.25). Figure 4.7(b) shows this cubic profile when looking at the interpulse spacing. Both the beginning and end pulses in the RGPO profile are spaced apart farther than the middle pulses. The power profile mirrors those shown by Figures 4.4(b) and 4.6(b), which will be explained further in Section 4.2.2. The final section for the RGPO range/delay profile discusses the general f-nomial case for the RGPO profile for function modeling.

4.2.1.4 f-nomial RGPO Profile Modeling. Table 4.1 shows how all three previously discussed polynomial cases show a functional relationship that can be expressed in a general case. It can be easily from looking back at final motion Equations (4.5), (4.34), and (4.49) that a generic relationship exists between the polynomial factor desired and the associated range equation. Looking back at the definitions given in (4.2)−(4.4), the position equation can be given in a general form of:

$$r(t) = \frac{C_m t^m}{m!} + \frac{C_{m-1} t^{(m-1)}}{(m-1)!} + \cdots C_1 t + C_0. \tag{4.74}$$

Table 4.1: RGPO Profile Velocity Constants

Variable	**Constant Velocity** ($f = 1$)	**Constant Acceleration** ($f = 2$)	**Linear Acceleration** ($f = 3$)
$V_{\min}$	$\frac{(R_{\max} - R_o)}{Tw}$	$\frac{(R_{\max} - 2R_o)}{Tw}$	$\frac{3(R_{\max} - R_o)}{2Tw}$
$V_{\max}$	$\frac{(R_{\max} - R_o)}{Tw}$	$\frac{(R_{\max})}{Tw}$	$\frac{(R_{\max} - R_o)}{Tw}$

Equation (4.74) for the generalized range-delay profile can be rewritten in summation notation as:

$$r(t) = \sum_{m=0}^{M=f} \frac{C_m t^m}{m!}, \tag{4.75}$$

where $r(t)$ has C_f coefficients and $C_0 = R_o$. Further determination of constants for (4.75) comes from exploring the other conditions highlighted in the three range-delay profiles examined. The first relationship explored is between $V_{\max}$ and $V_{\min}$ for the three f values explored previously. Table 4.1 shows the associated $V_{\max}$ and $V_{\min}$. Looking back at the parameters sets for all three profiles, the same three values make up the desired optimization parameters based on range. These parameters are:

$$\bar{u}_{\text{nom}} = [T_w, R_o, R_{\max}, f, JSR], \tag{4.76}$$

where all three are required to define the desired profiles. Specifying extra parameters, such as $A_{\min}$ or $V_{\min}$ serve to bound the problem space. These values specify limitations on C_0 to C_f for 4.75 and the associated values in 4.76. As shown from Table 4.1, the values for minimum and maximum velocities and accelerations can be defined in terms of the optimization parameters and serve as nonlinear constraints for the optimization routine. The profile selected f becomes an optimization parameter because each value represents a different representation from 4.74. The final component added to the optimization parameter set is the power profile, discussed in detail in the following section.

4.2.2 RGPO Jammer Signal Power. The signal power returned to each range bin is compared to the detection threshold level set at the output of the radar receiver. If the receiver output is large enough to exceed the set threshold, the radar declares a target is present. Where the tracking radar's threshold detector is set becomes dependent upon the environment space volume searched for targets. Referring back to Figure 4.2, the signal power received by the ECM suite from the radar transmitter is determined by the one-way free-space transmission equation [9], relating

(G_t), the transmitted power (P_t), the jammer's effective receive area ($A_{e_{\text{jam}}}$), and the one-way range (R) to the target, as:

$$P_O = \frac{P_t G_t A_{e_{\text{jam}}}}{4\pi R^2}. \tag{4.77}$$

The received power (P_O) from the radar transmitter is then reradiated back into space in various directions. The amount of power that is reradiated back in the direction of the tracking radar receiver (P_R) is based upon the target radar cross section (σ_t) and the return range distance as:

$$P_R = \frac{P_t G_t}{(4\pi R)^2} \frac{\sigma_t A_{e_{\text{jam}}}}{(4\pi R)^2}. \tag{4.78}$$

Equation (4.78) represents the product of power captured by the ECM suite in (4.77) and power density reradiated back at the tracking radar. Ideal conditions would ensure all power received at the jammer would be broadcast back at the radar receiver to effectively deceive the target tracker. Previous discussion from Section 2.2 shows that only a fraction of the transmitted power returns back to the target tracker due to the tracking radar aperture's physical size.

The decision criteria for determining whether or not a target exists, after receiving the signal, is determined from the probability of detecting a target with voltage signal V_r=$\sqrt{P_r}$. The expected signal V_e is detected with a probability density function [9]:

$$p_s(V_e) = \frac{V_e}{\psi_0} \exp\left(-\frac{V_e^2 - {V_r}^2}{2\Psi_0}\right) I_0\left(\frac{V_e V_r}{\Psi_0}\right), \tag{4.79}$$

where Ψ_0 represents the mean noise power for the radar system and V_e is the envelope detection voltage. The probability of detecting a target then having voltage V_e is expressed as:

$$P_d = \int_{V_t}^{\infty} p_s(V_e) dV_e. \tag{4.80}$$

Equation (4.80) [9] becomes vital in understanding that injected false targets by the jammer must observe this property to ensure the tracking radar does not dismiss

the false target. Furthermore, false alarms are avoided by tracking radars through carefully setting the automatic gain control (AGC) high enough to detect only true targets with minimal probability of miss. This issue becomes important to consider when implementing the RGPO profile to pull the tracker away using the injected target.

Understanding that the tracking radar tries to compensate for false targets, a closer look at (4.77) becomes necessary to determine the jammer power required to effectively deceive the tracking radar. The power seen at the receiver in Figure 4.1 is a product of the receiver power and the effective area of the jammer, which accounts for the tracker operational wavelength, the repeater jammer gain (G_{JR}), and the one-way polarization losses (L_p) between the tracking radar and the repeater jammer. This jammer input power can be expressed as:

$$P_{JR} = \frac{P_t G_t}{4\pi R^2} \frac{\lambda^2 G_{JR}}{(4\pi) L_p}. \tag{4.81}$$

The power received at the radar receiver can then be expressed as [25]

$$P_R = \frac{P_t {G_t}^2}{(4\pi R)^4} \frac{G_{JR} G_{JT} G_e \lambda^4}{{L_p}^2}, \tag{4.82}$$

where G_{JT} represents the output amplifier gain in (4.1) and G_e represents the repeater's antenna gain with all non-polarization losses accounted for. The jammer-to-signal ratio (JSR) for this scenario can then be seen as:

$$JSR = \frac{J}{S} = \frac{G_{JR} G_{JT} G_e \lambda^2}{4\pi \sigma_t {L_p}^2}, \tag{4.83}$$

which is independent of range, but accounts for all jammer induced gain and spatial polarization losses [25]. After calculating the necessary JSR to deceive the tracking radar, the repeater estimates the platform's own radar cross section and magnifies it to mask the true return: $\sigma_e = JSR\sigma_t$. The total repeater gain G_{REP} is then calculated

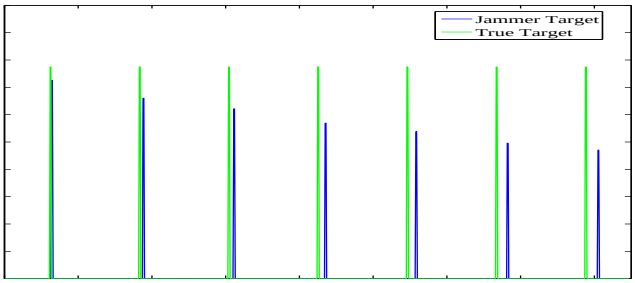


Figure 4.8: Jammer delay profile over multiple PRIs.

from the repeater RCS:

$$G_{\text{REP}} = G_{JR}G_{JT}G_e \tag{4.84}$$

$$= \frac{4\pi\sigma_e {L_p}^2}{\lambda^2}. \tag{4.85}$$

Substitution of (4.85) back into (4.82), gives the power at the receiver from the jammer as

$$P_R = \frac{P_T {G_t}^2 \lambda^2 \sigma_e}{(4\pi)^3 R^4}. \tag{4.86}$$

Equation (4.86) gives the victim radar's perception of the target once the jammer has magnified the pulse. This allows the jammer to spoof the tracker radar into thinking the target is larger than it actually is, but comes with physical tradeoffs that are explored in the next section. With the jammer power relationship established, the power profile for the delayed target signal must compensate for the added delay for accurate target deception. Figure 4.8 shows the relationship of delaying the false target a given distance from the true target without added power compensation. Yet the deception signal should include a power dissipation of $\frac{1}{R^2}$. An example of this dependency is shown in Figure 4.9. The profile shown in Figure 4.9 is:

$$\sigma_e = \frac{JSR\sigma_t}{(R - R_o)^2}. \tag{4.87}$$

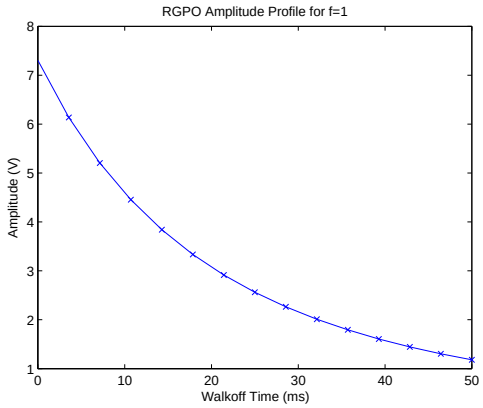


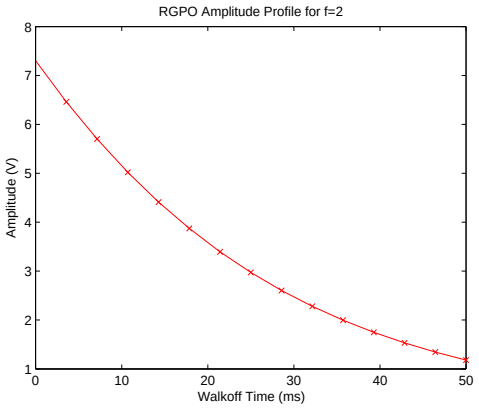
Figure 4.9: Amplitude delay profile.

Equation (4.87) must be modified because when $R = R_o$ the magnified RCS is infinity, which is unrealistic for engagement modelling. This condition is compensated for but shifting the decay function to the right. Applying a unit shift causes σ_e takes on the value of $JNS\sigma_t$ when $R = R_o$ as desired. By adding a simple shift, expressed by

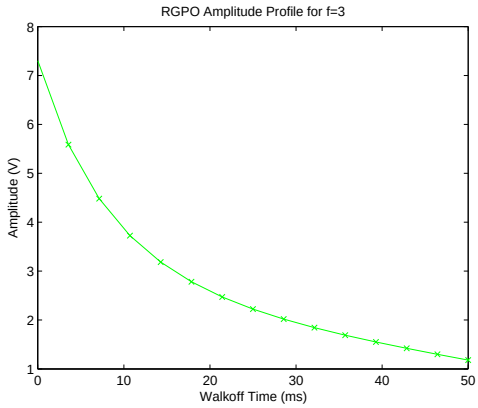
$$\sigma_e = \frac{JSR\sigma_t}{(R - R_o + 1)^2}, \tag{4.88}$$

gives the desired values for the RGPO power profile to deceive the radar. Figure 4.10 shows the amplitude delay profiles corresponding to Figures 4.6 and 4.7. Although each profile has the same amplitude decay rate, the pulse spacing is different between the Figures 4.6 and 4.7. These figures show that both the initial jammer signal amplitude A_{j_o} and minimum jammer signal amplitude $A_{j_{\min}}$ modeled for the jammer signal depend on the walk-off distance and not the actual pulses in the signal profile. Equation (4.88) reveals the limitations to the desired RGPO ECM waveform, which is explored further in next section.

4.2.3 Physical Limitations. Figure 4.10 shows the effects of inducing range delays on the true target return. The power profile modeled shows that inducing target delays comes at the cost of decreased power returned to the tracking radar. By adding the extra range distance, shown in Figure 4.10 as a time-delay, the received



(a) Amplitude delay profiles for RGPO signal ($f = 2$).



(b) Amplitude delay profile for RGPO signal ($f = 3$).

Figure 4.10: RGPO amplitude delay profiles from Figures 4.6 and 4.7.

power subsequently decreases. This power decrease is proportional to $\frac{1}{R^4}$. When establishing the appropriate JSR level to mask the true target return, the jammer must consider the true target return power level and the walk-off time when setting the power profile. If the false target is not set high enough, like in Figure 4.8, the target tracker never locks onto the false target. Conversely, if the power level received is too large, the tracker's ECCM detects that the power spike is from a jammer and negates the false target. Further consideration to walk-time also defines the minimum power required to continually mask the target. As shown from Figures 4.4 to 4.7, the walk-off distance is so large that the power drops below the minimum level desired to mask the true target. Careful consideration must be given then to ensure that the jammer peak power falls within these limits. Another physical limitation that must be considered is the relative motion of the false target to the true return. For example, in the Lab-Volt™ radar system specifications, the range span for the Lab-Volt™ radar covers a maximum of 7.2m and the associated jammer pull-off distance equals 0.512m [4,23]. If the jammer's pull-off rate is $5\frac{cm}{s}$ [4], the Lab-Volt™ tracker may not be able to follow the jammer's false target. When the target tracker can no longer track the false target, the radar tracking loop returns to the last known true target return stored in the tracker memory [26]. While some military tracking radar algorithms

can handle up to nine times gravitational acceleration ($9g$), typical modeled pull-off rates are on the order of three times the acceleration due to gravity, ($3g$) [26]. These limitations are designed into the Lab-Volt™ training system. Figure 4.3 shows the range pull-off profile typical to RGPO jammers like the Lab-Volt™ training system jammer, which are modeled using a constant velocity RGPO signal. In comparison, if equation (4.44) were used to model the RGPO signal for the Lab-Volt™ Jammer Pod, the total delay would be 0.998 m. The change in range-delay profile would increase the Lab-Volt™ walk-off rate by 48.7%. The subsequent acceleration rate from changing the RGPO model is $1.175\frac{m}{s^2}$. Although this acceleration falls within previously discussed acceleration limitations, the Jammer Pod could not handle the power increase necessary to inject the extra target distance. Although the initial design phase was to simulate the Lab-Volt™ jammer pod, the designed MATLAB® RGPO jammer has the flexibility to model the f-nomial function defined in Section 4.2.1.4 or other desired models for a genetic algorithm to optimize.

4.2.4 MATLAB® Implementation. Figure 4.11 shows the block diagram of the RGPO simulator developed in MATLAB [26]. In implementing the RGPO circuit, the memory circuit from Figure 4.1 is removed and the program circuit and accompanying software control the number of delay steps to the range profile. The input and output amps are tuned to model the desired JSR. The *pulsepower* MATLAB code developed represents the input/output amplification modelled in equation (4.88) to depict the power decay as shown in Figure 4.8. The signal memory block shown in Figure 4.11 is implemented through the software counter defined in the RGPO main program. Any necessary changes to the pulse shape are implemented through the pulse generation subfunction to induce necessary phase delays. The counter (m) runs through the various steps to the number of delays, given in 4.2. The delay code shows how the counter is implemented to produce the delayed pulse. For this model, σ_t is calculated based upon the target mounted on the Lab-Volt™ jammer pod. These various radar cross section (RCS) values are programmed based upon a lookup table

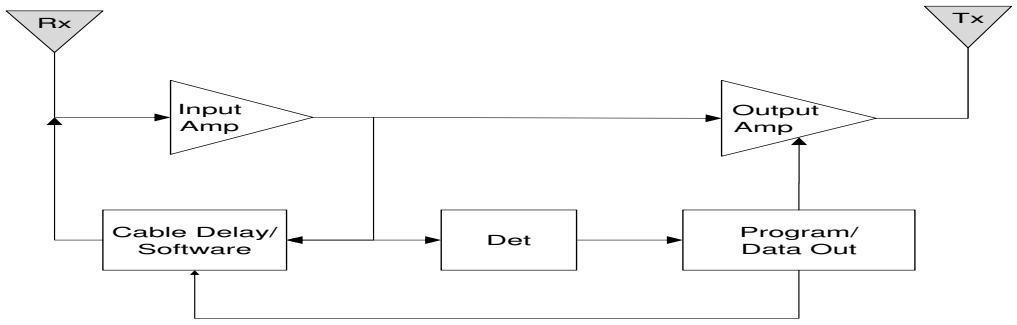
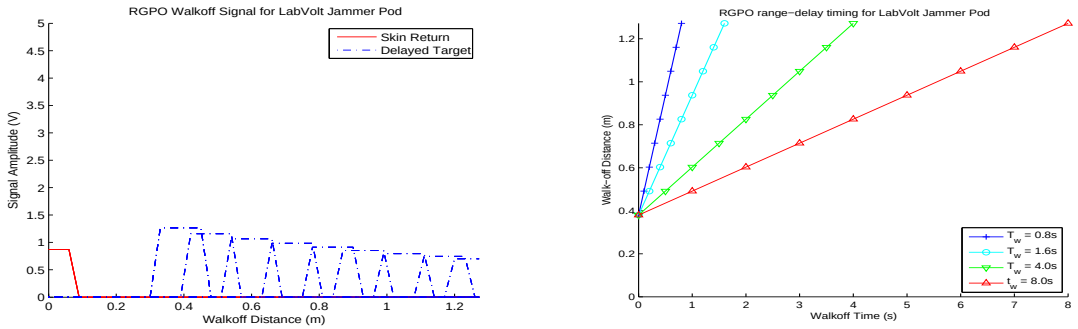


Figure 4.11: Lab-Volt™ RGPO implementation block diagram, adapted from [27].

derived from the mathematical formulas for representing these targets [29], contained in radar cross-section code. These RCS values are considered Swerling Case 0, or non-fluctuating targets. Future implementation of a generic jammer can be implemented using either known RCS values or calculations based upon defined geometric shapes, different Swerling models, and the materials used. Upon determining the associated σ_t for transmission, the necessary jammer transmission power is developed through (4.86) and is transmitted into the environment. Developing the RGPO jammer in MATLAB required accurate depiction of the Lab-Volt™ Jammer Pod and the desired operation modes. Table 4.2 gives the modeling parameters for the RGPO waveform. The notable difference between the Lab-Volt™ Jammer Pod and the mathematical development in Section 4.2.1 is the number of delays used to model the profile shown in 4.3. The Lab-Volt™ jammer uses software to set $M_{\text{J}} = 8$ and using equation (4.21)

Table 4.2: Lab-Volt™ Jammer Pod RGPO Parameters [23].

Parameter	**Value**
Distance between delays (cm)	6.4
Initial delay $\delta_i, (cm)$	38
Maximum Input Power (dBm)	10
Number of delays M_{J}	8
Number of RGPO rates	4
RGPO rates (s)	0.8, 1.6, 4.0, and 8.0



(a) Lab-Volt™ RGPO range/delay profile.

(b) Lab-Volt™ RGPO signal.

Figure 4.12: Lab-Volt™RGPO constant velocity profile

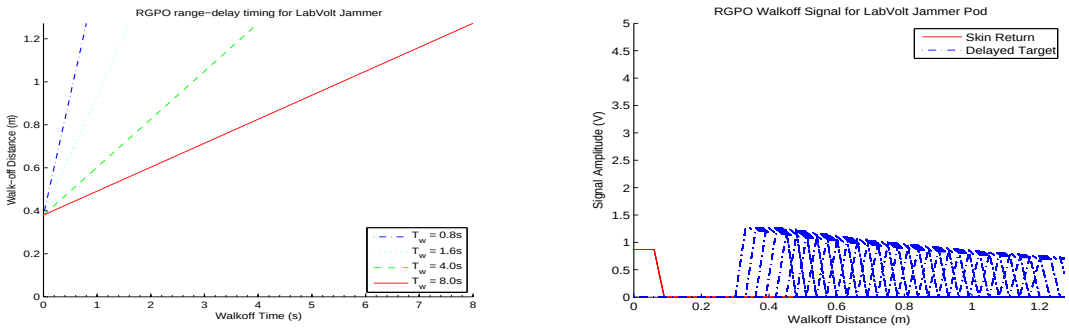
with $R_{\max}, R_o$, and T_w from Table 4.2 results in the RGPO signal.

$$r(w) = .512\left(\frac{m}{8}\right) + .38 \tag{4.89}$$

Figure 4.12 shows the Lab-Volt™ Jammer signal as simulated by the using delay implementation code using Equation (4.89). The range-delay plots in Figure 4.12(a) show the change in pull-off rates due to the walk-off time desired. The 8 delay points correspond to the Lab-Volt™ Jammer Pod specifications of 6.4cm distance between delays and the RGPO signal shown in Figure 4.12(b) validates this spacing. The sole modification required was for incrementing the counter variable m. Equation (4.90) shows the modification to the counter variable.

$$m_l = \left\lfloor w\frac{8T_p}{T_w} \right\rfloor \tag{4.90}$$

Equation (4.90) allows the program counter to broadcast the number of pulses necessary for operators to visualize the waveform while maintaining the necessary walk-off distance and walk-off time to accurately portray the signal. Figure 4.12(a) shows similar delay distances as depicted in Figure 4.13, which uses the continuous time function equation, defined by (4.44). Comparison between Figures 4.12 and 4.13 show the same walk-off distances and amplitude modulation for the two signals. Where these two range profiles differ is that equation (4.90) uses a defined number of PRIs by the



(a) Lab-Volt™ RGPO range/delay profile.

(b) Lab-Volt™ RGPO signal.

Figure 4.13: Non-stepped Lab-Volt™RGPO constant velocity profile

jammer software and Figure 4.13 uses the defined PRIs based upon equation (4.20). The number of pulses in the walk-off profiles shown in Figure 4.13 range from 228 for the 0.8s walk-off time to 2286 pulses for the 8.0s walk-off time.

4.2.5 RGPO Development Summary. The MATLAB® simulation and the mathematical representation of the Lab-Volt™ jammer pod shows that the RGPO general case developed in Section 4.2.1 can accurately represent any jammer used for modeling. Furthermore, this flexibility highlights the concept of modular jammer design for the HILS architecture given in Section 3.1. The development framework for the RGPO case lends itself to development of the VGPO profile model. The next section develops the VGPO model using core pieces of the RGPO model, but using the STAP model to process the necessary Doppler shifts.

4.3 VGPO Techniques

As discussed briefly in Section 1.1, the VGPO jammer generates a false target with an induced frequency offset relative to the true target return. The induced frequency offset on the returned pulses varies as a function of time to show the target accelerating away from the victim radar. The associated time history for the VGPO signal is repeated such that the victim's radar range rate tracker is pulled away from the true target return, causing the tracker to break lock [30]. The frequency modu-

lation waveform used to induce the false Doppler returns onto the repeated jammer signal then becomes a critical parameter for the deception jammer.

The VGPO waveform development is presented in three parts. The first section looks at the VGPO basic signal, understanding how the imposed Doppler shifts add the velocity component to the range/velocity tracking done by modern radar systems. The second section looks at how the Doppler shift can be imposed using the Ward STAP model [31]. By using the STAP model to implement the VGPO signal, coordinated RGPO and VGPO is easily accomplished and may improve efficiency during optimization because of the matrix framework [31]. The final section looks at the developed VGPO signal model and applies it towards a known radar system. From the implementation of the VGPO signal model, the velocity changes can be mimicked with either a pull-off profile or a pull-in (VGPI) profile and add additional signal models to the ECM library.

4.3.1 VGPO Mathematical Background. In modeling the deception jammer waveform given in Equation (4.91), A_j represents the signal jammer voltage amplitude, ω_J is the normalized Doppler frequency and ϕ_J is the phase-shift induced by the deception jammer. Equation (4.91) gives the deception jammer signal broadcast, knowing the true target skin return, as [25]:

$$S_J = A_j \cos(w_J t + \phi_J). \tag{4.91}$$

The normalized frequency offset imposed by the jammer comes with two limitations that must be understood to successfully conduct VGPO jamming. First is that ω_J must fall within the Doppler passband of the initial target. While broadcasting with the initial frequency offset, enough time must be given such that the victim's AGC adjusts to the false target. As with the generalized RGPO waveform developed in [1] and previous discussion, the target skin return must be masked and amplified such that the AGC threshold is above the true target return. Second, in trying to deceive the victim tracker, it is important to understand the victim radar's Doppler resolution

capabilities. To accurately use Equation (4.91) to walk off the true target return, ϕ_J must change in increments that the victim radar can detect, with maximum frequency offsets ranging between $5 - 50$ kHz [25]. Another important aspect in modeling the phase changes requires consideration of the associated acceleration changes that come with frequency modulation. Most Doppler radars associate differential velocity changes as target acceleration. With $a_t = \frac{\delta}{\delta t} v_t$, the velocity tracker checks for unusual accelerations derived from either unusual jumps in velocity or values that exceed the target's expected performance specifications. This leads to constraints on ϕ_J modeled in the following manner [25]:

$$\phi_j(t) = \frac{\partial}{\partial t} f_d(t) \tag{4.92}$$

$$= \frac{\pi}{2^{b_J - 1}} \tag{4.93}$$

where b_J represents the number of discrete phase bits utilized by the jammer to add the frequency shift to the output signal [25]. This parameter serves as an important variable in $\bar{u}_c$ for the VGPO signal. The number of phase bits determines how accurate the velocity signal walks away from the true target. Previous discussion from Nunez *et al.* suggests that a linear-ramp frequency offset is ideal, giving a constant acceleration [1] in modeling RGPO. While this handles some cases, changes in the mathematical expression to represent other polynomial expressions could give the deception signal the realistic representation of maneuvering in an attempt to evade the radar tracker. While ensuring that the appropriate frequency offsets are applied, the associated voltages applied are also important. The deception jammer power injected into the target environment can be modeled knowing that the jammer can accurately detect the signal power from the victim radar. The power profile for the VGPO signal is identical to the RGPO power profile discussed in section 4.2.2. Through prior knowledge of the victim radar operational frequency f_o, transmitter power P_t, and the target range to the victim radar R, the jammer power P_J equates to (4.81). This

radar range equation modification gives the desired signal power necessary to deceive the tracking radar with an injected false target.

The target range can be estimated from the pulse repetition frequency used by the victim radar and usually calculated through the use of Digital Radio-Frequency Memory (DRFM) chips within the repeater jammer. DRFM chips provide a memory capability independent of storage time that allows coherence of captured signals, preventing signal deterioration as time delays are introduced. Through knowing designed jammer characteristics, the amplitude roll-off function is similar to that discussed in [1], compensating for changes in range along with desired jammer-to-signal ratios.

For signal optimization, the linear VGPO pull-off signal is a function of the relative velocity between the tracking radar platform and the jammer platform. The relative Doppler frequency is:

$$f_{\text{rel}} = \frac{2(V_a - V_t)}{\lambda_o}, \tag{4.94}$$

where V_a is the jammer platform's velocity and V_t is the tracking radar platform's velocity. The normalized Doppler frequency is desired for false-target placement in the STAP data-cube, which can be determined by:

$$\hat{f}_{\text{J}} = \frac{f_{\text{rel}}}{f_{PRF}} - \left\lfloor \frac{f_{\text{rel}}}{f_{PRF}} \right\rfloor. \tag{4.95}$$

The number of pulses M_{JV} required to step through the tracking radar's velocity gates is:

$$M_{JV} = M\delta V = (\hat{f}_{\text{J}_{\max}} - \hat{f}_{\text{J}})M, \tag{4.96}$$

where $\hat{f}_{\mathrm{J}_{\max}}$ is the normalized value of the maximum walk-off velocity. From (4.96), b_J is determined after substitution into (4.93):

$$M\delta V = \frac{\pi}{2^{b_J-1}} \tag{4.97}$$

$$b_J = \left\lceil \log_2 \left(\frac{\pi}{(\hat{f}_{\mathrm{J}_{\max}} - \hat{f}_{\mathrm{J}})M} \right) \right\rceil + 1. \tag{4.98}$$

The ceiling function is required for b_J to determine the whole number of bits to implement the phase shift. Furthermore, one extra bit is added based upon the 2s-complement representation for the bit-stream. This representation allows for count-down use in possible Velocity Gate Pull-In (VGPI) implementation, which the model implemented uses 6 bits. The amplitude scaling necessary for the VGPO waveform is the power profile given by (4.88). The mathematical function $\Phi(u)$ for jammer implementation and subsequent waveform optimization is then expressed as:

$$\Phi(u) = A_j \cos(w_J t + \phi_J), \tag{4.99}$$

$$\phi_J = (\hat{f}_{\mathrm{J}_{\max}} - \hat{f}_{\mathrm{J}}) \frac{m}{M_{JV}} f_{\mathrm{PRF}}, \tag{4.100}$$

where

$$m = 1 : M_{JV}, \tag{4.101}$$

sequentially stepping the phase changes through the desired maximum velocity and M_{JV} is bounded both by the modelled acceleration rate and the tracking radar's CPI. The parameter set $\bar{u}$ necessary for modeling $\Phi(u)$ for the VGPO signal is:

$$\bar{u} = [T_w, R_o, R_{\max}, JSR, f, M_{JV}]. \tag{4.102}$$

M_{JV} becomes an essential parameter because of the jammer's dependence on the radar system's CPI length for STAP implementation. The jammer system hardware necessitates that the number of steps in pulling off the velocity gate be on the order of $M_{JV} = 2^{b_J-1}$ for adequate signal representation. Development of $\Phi(u)$ and $\bar{u}$

Table 4.3: Radar Model Parameters.

Parameter	**Value**
Aircraft Velocity - V_a,($\frac{\mathrm{m}}{\mathrm{s}}$)	200
Aircraft Altitude - h_a,(m)	3048
Azimuth Channels - N	8
Elevation Channels - P	8
Azimuth Element Spacing - dx, (m)	$\frac{\lambda}{2}$
Elevation Element Spacing - dz, (m)	$\frac{\lambda}{2}$
Transmit Power - P_t, (kW)	200
Coherent Pulse Integration (CPI)	64
Boresight Angle - $\bar{\theta}_o, \bar{\phi}_o$	$(0°, 0°)$
Operating Frequency - f_o, (GHz)	1.24
Pulse Repetition Frequency - f_r, (Hz)	1984
Pulse Width - τ (μs)	10
Radar Noise Figure - (dB)	3
Radar Range Resolution - ∂R (m)	151
Radar Velocity Resolution - ∂V, ($\frac{\mathrm{m}}{\mathrm{s}}$)	3.75

allows for HILS optimization, as shown in [21], which is assisted through a STAP implementation of the VGPO waveform. The next section further develops the STAP model for use in VGPO waveform development.

4.3.2 STAP Developed VGPO Signal. The model parameters from Table 4.3 come from the multi-channel airborne radar measurement (MCARM) system [32], operating as a pulse-Doppler radar mounted on an airborne platform moving with a constant speed (V_a). The radar antenna is a uniformly spaced linear array with N azimuth elements by P elevation elements [31]. The transmitted signal generated by the MATLAB model data is assumed as a narrow-band signal that detects targets in the far field. Through these assumptions, the voltage received along the array, at the nth row element and the pth column element, is a function of the angle of arrival with respect to θ and ϕ, the target Doppler velocity, v_d, and the normalized phase difference across the array, $\frac{2\pi d}{\lambda}$. This signal representation using complex envelope notation is

$$s_q(t) = \alpha(t) \exp\left(j2\pi\left(\frac{qd}{\lambda}\sin\theta + v_d t\right)\right) \tag{4.103}$$

where $\alpha(t)$ represents the down-sampled voltage complex amplitude, derived from (2.1), and the subscript q denotes the number sensor, referenced as $[n, p]$ in the array [31]. With this representation, the mutual coupling effects along the array are ignored and the target return produces only a linear phase difference along the array. Figure 4.14 illustrates the data cube collected during the receiver CPI, including how the radar aperture elements map directly to the data cube [33]. Once the phase compensations have been made, the data can be coherently added together through the coherent processing interval (CPI) of pulses collected. Each of these M pulses within the CPI are then sampled at a given range gate, R, along the total number of sensors. Furthermore, the Doppler and angular information can be resolved through understanding that the array's linear spacing allows for discrete shifts between sensors over all frequencies. Through development of a steering vector $\overline{v}$, phase shifts along the sensors can be corrected to allow coherent data processing. The steering vector $\overline{v}$ then takes on the following form:

$$\overline{v}(t) \equiv [1 \; v \; v^2 \; \cdots \; v^{(N-1)}]^T, \tag{4.104}$$

where each element of $\overline{v}$ represents the exponential term described in (4.103). When applying $\alpha(t)$ to (4.104), the complex voltage received, the resultant signal is $\overline{s}(t) = \overline{v}(t)^H \alpha(t)$. The radar aperture shown in Figure 4.14 leads to the construction of three column vectors representing the radar pulses collected across the aperture from a given location in space. The temporal steering vector ($\overline{b}(t)$), denotes the M-dimensional Doppler returns to the radar aperture, while the other two denotes the number of azimuth elements, ($\overline{a}(t)$), and elevation elements ($\overline{e}(t)$), shown in Figure 4.14, which are expressed as

$$\overline{e}(t) = [1 \; \exp(j2\pi\overline{\omega}_{te}) \; \cdots \; \exp(j2\pi(P-1)\overline{\omega}_{te}]^T, \tag{4.105}$$

$$\overline{b}(t) = [1 \; \exp(j2\pi\nu_t) \; \cdots \; \exp(j2\pi(M-1)\nu_t]^T, \tag{4.106}$$

$$\overline{a}(t) = [1 \; \exp(j2\pi\overline{\omega}_{ta}) \; \cdots \; \exp(j2\pi(N-1)\overline{\omega}_{ta}]^T \tag{4.107}$$

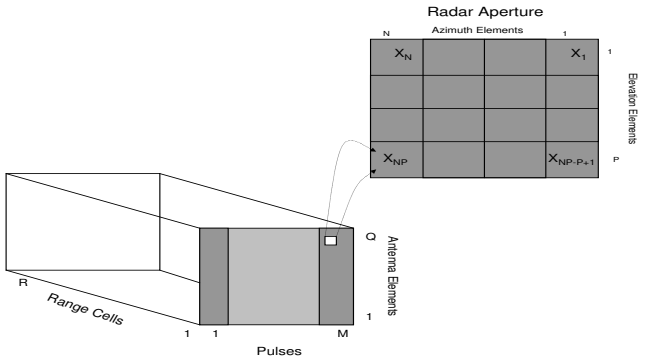


Figure 4.14: Illustration of STAP 3D data cube construction [33].

with

$$\nu_t = \frac{f_t}{f_r}, \tag{4.108}$$

$$f_t = \text{target Doppler velocity}, \tag{4.109}$$

$$\overline{\omega}_{ta} = \frac{d}{\lambda}\sin(\theta_t), \tag{4.110}$$

$$\overline{\omega}_{te} = \frac{d}{\lambda}\sin(\phi_t). \tag{4.111}$$

Formation of (4.104) comes from combining (4.105)-(4.107) in the following operation,

$$\overline{v} = \overline{e} \otimes \overline{b} \otimes \overline{a} \tag{4.112}$$

where $\otimes$ represents the Kronecker product of the two vectors. This operation creates the spatial 2-D FFT across the array over all possible look angles in (θ_t, ϕ_t), summing the returned signal across the entire array. From (4.112), the resulting vector creates a 3-D space-time snapshot for a particular range. This accounts for the aperture's collected signal in both (θ_t, ϕ_t), but does not consider environmental noise and clutter returns, prior to understanding target detection.

4.3.3 MATLAB® Jammer Development. Using STAP with the VGPO jammer integrates the mathematical equations from the previous section into the MATLAB model from previous work [31]. Table 4.4 lists the target parameters that are

Table 4.4: Target Characteristics.

Parameter	Value
Target Range - R,(km)	37.8-75.6
Target Velocity - V_t, $\frac{\mathrm{m}}{\mathrm{s}}$	-200 - 200
Target Radar Cross Section - σ_t	5
Jammer Type -	Deception
Jammer Mode -	Uncoordinated VGPO
Jammer-to-Signal Ratio - (dB)	-18

stored in vector form for use in the model. The σ_t value used represents a medium-sized fighter, similar to the RCS of a third-generation aircraft, derived from Table 2.1 of Skolnik [9]. The injected target is given a random velocity and distance from the victim radar, from a uniform random number, in the following manner:

$$\begin{aligned} R_t &= U(250, 500), \\ V_t &= U(-200, 200). \end{aligned}$$

R_t is given a distance of at least $250\partial R$ to represent a target somewhere in the far field of the tracking radar. The velocity component, given in $\frac{m}{s}$, uses a uniform random number generator to determine the velocity sign while another determines the velocity magnitude. The modeled targets fall under Swerling Case 0 where there are no RCS fluctuations due to changes in pulses or scan-to-scan changes. The targets are injected into a noisy environment, but clutter is ignored in the environment only for decreasing processing time. The target generating function injects the target with the desired velocity in the STAP model according to previously developed work [34]. Once the targets are injected, the VGPO ECM function applies the necessary Doppler shifts and amplitude changes necessary to represent a real jammer engagement. Prior to invoking the VGPO ECM function, the system calculates a few important parameters. First, the jammer determines the Doppler resolution as if communication took place between the Radar Warning Receiver (RWR) and the deception jammer. This link passes known Doppler tolerances of the victim system such that the linear progression could be accurately modeled. Next, the jammer calculates the true target return's

actual Doppler frequency f_{rel}, noting possible aliasing that could occur in the victim radar, determined in equation (4.95). From determining the appropriate Doppler bin for the target return, the jammer processor then calculates the number of bins necessary to pull off the velocity towards the victim's maximum detectable velocity [34]. This model assumes the linear case discussed in the previous section, although the polynomial case can be implemented through further development of how the jammer selected the number of bins to use. Finally, the associated direction of walk-off is determined through the sign of the target Doppler. For example, if the target velocity is $-100\frac{m}{s}$, the target should move towards the minimum velocity and not through zero towards the positive maximum. Placing the masked target return in Doppler comes from manipulating (4.91) with respect to applying the normalized Doppler with the steering vector, expressed in (4.112) [35]:

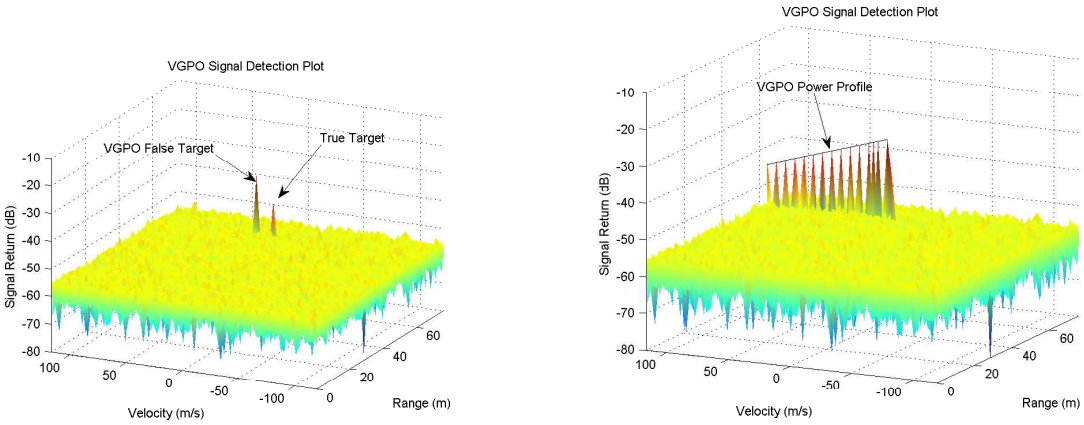
$$\bar{b} = \exp(j2\pi(f_{rel} + \phi_J)), \tag{4.113}$$

where both f_d and ϕ_J are normalized by the victim's PRF for the appropriate frequency shift. This shift represents the specific increase in velocity at a particular time in the radar.

The final component to masking the true target return implements a_t for the jammer signal output. While equation (4.88) gives the necessary power for the jammer required to be seen at the victim radar, this model parallels target modeling with changes in radar cross-section to induce necessary amplitude changes. To apply cross-sectional changes to (4.88), the collective gain for the jammer and receiver is defined by (4.88) discussed in Section 4.2.2, where σ_e is the magnified radar cross section for the deception jammer. Implementation of σ_e in (4.88) in the VGPO walkoff script is a function of the desired jammer-to-signal (JSR) ratio, the target RCS and the linear power profile given in (4.88). The magnified deception jammer RCS, σ_e, is amended for the initial target masking to prevent unnecessarily large initial returns at the radar receiver. The jammer model processes the received pulse return and

magnifies the signal by the desired JSR. In the initial return case, the jammer signal is intended to increase the AGC. If this gain is not carefully monitored, electronic counter-countermeasure (ECCM) flags would trip due to a power spike and reject the jammer signals. This condition is prevented by the conditional statement developed in the jammer for the initial target return. Once placed in doppler and given the associated jammer voltage, the final radar target data operation applies the voltage signal $\sqrt{P_r}$ to $\bar{v}$ at the r^{th} range bin. This target data model assumes targets fall only in a single range bin and that there are no correlation terms between either target's RCS.

4.3.4 VGPO/VGPI Simulation and Results. Figure 4.15(a) shows the final result from the deception jammer mixed with the actual target return at the victim receiver. Figure 4.15(a) comes from a specific snapshot of the radar signal processor after receiving the collection of data from the target environment. The color map imposed on the figures in this section show relative power at the victim radar receiver after STAP. The chosen color scheme provides ease of visual target identification. The target environment is created as a movie developed to depict near real-time processing by the radar of the jammer and target signals received from the environment. These movies are generated through looping through the jammer function for W snapshots necessary to walk-off the velocity gate through $V_{\max}$. In developing the signal in this manner, the VGPO ECM format is formulated to coincide with both the RGPO model developed in Section 4.2.1 and the scoring function described in Section 3.2.5. The VGPO signal illustrated in Figures 4.15 and 4.16 are considered uncoordinated VGPO/VGPI signals. This suggests that the target movement is staying in the same range bin but is increasing or decreasing its velocity relative to the victim radar. Each jammer created signal is added to the target and noise signals, simulating the collection of all signals at the radar antenna. Once the entire suite of jammer snapshots are created, the radar signal processor MATLAB function manipulates the data in succession to create the picture seen in the STAP models for one instant in time. All the

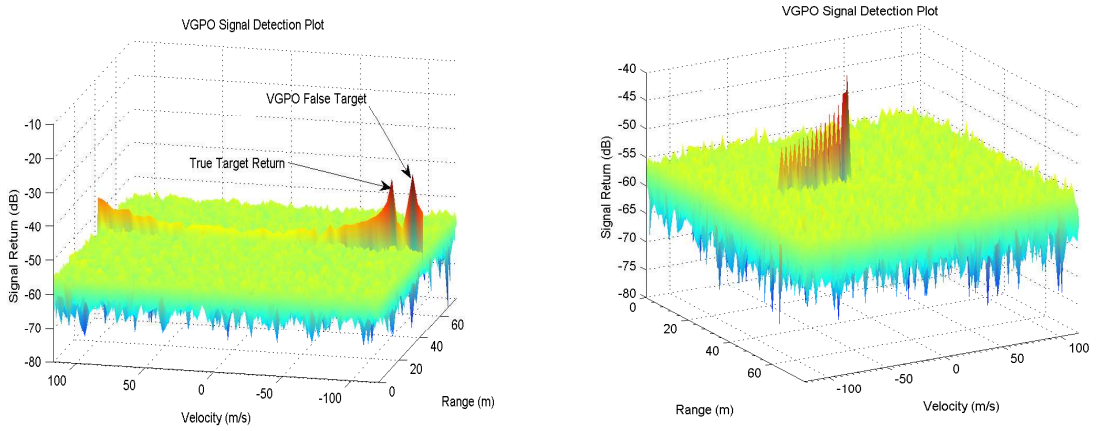


(a) VGPO model example.

(b) VGPO pull-off profile.

Figure 4.15: VGPO jammer signal simulation.

snapshots are processed in sequence to model how the jammer is pulling the target off in velocity. Figure 4.16(a) shows the VGPO jammer based on a target with a positive velocity. This graphic mirrors the VGPI jammer signal shown in Figure 4.15(a) created from the MATLAB code to represent the target deceiving the radar by moving closer to the victim radar. These snapshots come from the movie depicting the VGPO jammer against the target injected into the engagement. Figure 4.15(b) shows the summation of false targets over time in the power profile. Each peak represents the returned jammer power at a given time instance. Figure 4.16(b) shows the similar jammer power profile for the positive power profile change. Both 4.15(b) and 4.16(b) show the peak power at time zero, where the jammer frequency repeats the target return and terminates at the maximum measurable frequency change. The power change rates reflect the linear slope discussed in Schleher [25]. Figures 4.16(a) and 4.16(b) show the case where the target aircraft is moving away from the victim radar receiver. The target placement function, which serves as jammer main program, can be modified to equate non-linear roll-off functions by adding a polynomial function to the σ_e calculations. Any power modifications to the VGPO signal should consider the range power changes modeled in section 4.2.2. The roll-off profile changes follow dis-



(a) VGPI model example

(b) VGPI pull-off profile for Figure 4.15(a)

Figure 4.16: VGPI jammer signal simulation.

cussion in both Schleher for linear VGPO and in [1] regarding the generalized pull-off function.

One area not investigated at this point in model development is non-linear frequency modulation for the VGPO jammer. Careful $\bar{b}$ modification for non-linear changes of ϕ_J would equate to positive or negative accelerations. These modifications are done through understanding the general RGPO/VGPO profile development. Furthermore, the VGPO profile can be found through the derivative of the generic RGPO profile expressed in (4.75), allowing for seamless integration. Through the STAP framework, the induced false target distance comes from the previously calculated velocity change and the range profile. The profile depicted in Figure 4.17 shows a linear velocity profile change of $50\frac{m}{s}$ through the engagement. The target is placed at $25\delta R$, which enables the full profile depiction in Figure 4.17. The generalized RGPO jammer signal incorporated into this model illustrates the coordinated RGPO/VGPO jamming used against tracking radars with advanced ECCM suites. During this implementation, coordination is necessary between range bin placement and Doppler frequency addition to ensure that non-realistic velocities and accelerations are shown in range-Doppler map.

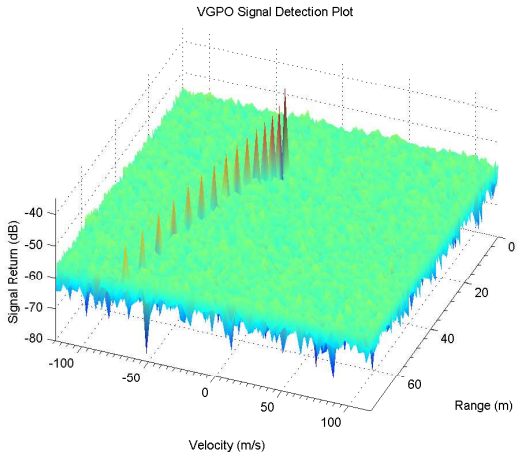


Figure 4.17: Coordinated RGPO/VGPO implemented in STAP.

4.4 ECM Waveform Modelling Summary

Mathematical development of the RGPO and VGPO waveforms creates the primary building blocks in the ECM library. Through understanding how both signals are created, proper MATLAB® encoding was established to enable the GA to optimize either based upon the defined parameter set. The generalized RGPO waveform can be incorporated in either a stand-alone RGPO model or used with a VGPO signal for advanced ECCM techniques. As future development continues, the mathematical representations for each waveform are stored in the ECM library for use by any given jammer. Development of two distinct jammer platforms shows flexibility of implementation. As long as the specified jammer parameters can handle the range of ECM library functions, the operator has freedom to define a particular ECM waveform for optimization. The next chapter explores the GA implementation of optimizing these two waveforms against a generic radar system.

V. Genetic Algorithm Optimization of RGPO

This chapter explores GA for developing ECM waveforms. The MATLAB® Genetic Algorithm and Direct Search toolbox serves as easy-to-use optimization routine that can be easily integrated with the defined HILS architecture. Previous work by Lamont and Landis shows that the ECM waveforms can be optimized, but more work is required to separate the optimization routine from the ECM signal generation. Demonstration of the separable components of optimization routine and ECM waveforms is shown with the RGPO waveforms. Future work from this research seeks to investigate other optimization routines besides GA in this architecture. Comparison between various optimization methods will determine which efficiently develops ECM waveforms for the HILS architecture. After efficiency studies have been done on waveform optimization, other ECM waveforms require development to further expand the ECM library for the HILS architecture.

5.1 MATLAB® GA Implementation

This first section builds upon work by Lamont and Landis (discussed in Section 2.4.3 [15]), which explored the development of Range Gate Pull-off countermeasure waveforms via the MATLAB® genetic algorithm toolbox. To better understand how to apply GA to ECM technique generation, a simple minimization problem is examined.

5.1.1 Analytical Bowl Minimization. The 3-D bowl provides a simple function with a known solution if the function Z describes an elliptical bowl located at $(x_o, y_o) = (5, 4)$:

$$\begin{aligned} Z &= a(x - x_o)^2 + b(y - y_o)^2, && (5.1) \\ &= 2(x - 5)^2 + 3(y - 4)^2, && (5.2) \end{aligned}$$

then the known minimum can be found by calculating the function's gradient [36]:

$$\min Z(x, y) = \frac{\partial z}{\partial x} Z(x, y) + \frac{\partial z}{\partial y} Z(x, y) \tag{5.3}$$

where

$$\frac{\partial x}{\partial z}Z(x,y) = 0, \tag{5.4}$$
$$\frac{\partial y}{\partial z}Z(x,y) = 0. \tag{5.5}$$

Applying equations (5.4) and (5.5) to (5.2) gives the solution:

$$\frac{\partial x}{\partial z} = 4(x-5), \tag{5.6}$$
$$x = 5, \tag{5.7}$$

and

$$\frac{\partial y}{\partial z} = 6(y-4) \tag{5.8}$$
$$y = 4. \tag{5.9}$$

The constants a and b define curvature of the bowl, but further exploration of this function is required to determine whether or not the solution is a local minimum for this function. The Second Partials test [36] is required to resolve this question. This test states that if $f(x,y)$ has a continuous second partial derivatives in the neighborhood of the minimum, (x_o, y_o), that the test [36]:

$$D(x_o,y_o) = \frac{\partial^2 x}{\partial z^2}f(x_o,y_o)\frac{\partial^2 y}{\partial z^2}f(x_o,y_o) - \frac{\partial x \partial y}{\partial z^2}f(x_o,y_o)^2 \tag{5.10}$$

Where if:

1. $D > 0$ and $\frac{\partial^2 x}{\partial z^2}f(x_o,y_o) < 0$, $f(x_o,y_o)$ is a local maximum value.
2. $D > 0$ and $\frac{\partial^2 x}{\partial z^2}f(x_o,y_o) > 0$, $f(x_o,y_o)$ is a local minimum value.
3. $D < 0$,$f(x_o,y_o)$ is not an extreme value.
4. $D = 0$, the test is inconclusive.

Following this test, $\frac{\partial^2 x}{\partial z^2} f(x_o, y_o) = 4$ and $\frac{\partial^2 y}{\partial z^2} f(x_o, y_o) = 6$. Evaluating $D(5, 4)$ results in:

$$D(5, 4) = (4)(6) - (4(0) - 6(0))^2 = 24, \tag{5.11}$$

which verifies that the point $(5, 4)$ represents the local minimum based upon the Second Partials test. While this solution is trivial due to a general understanding of calculus, this problem serves as a good test for the MATLAB® GA function. A known solution for a simplistic mathematical function allows exploration of the GA function's inputs and outputs, along with knowledge of parameter passing and necessary formatting for a desired fitness function.

5.1.2 MATLAB® GA Tool Definitions. The MATLAB® GA toolbox, given as `ga.m` or the GUI interface `gatool.m`, finds the local constrained minimum in a defined objective function [37]. The GA accepts a $1 \times N$ vector argument set as input to the objective function which is expressed for the minimization problem. Table 5.1 gives the various inputs for optimization using the GA toolbox. The variables listed in Table 5.1 are listed in order of their inclusion to running the GA function tool. In specifying Equation (5.2) as a fitness function for MATLAB®, the variables must be listed in a column vector, i.e. $x(1) = x$ and $x(2) = y$. The options variable

Table 5.1: GA Optimization Parameters [37]

Variable	**Definition**
fitnessfcn	Fitness function to evaluate
nvars	Number of variables in fitness function to optimize
options	Options structure for GA tool
Aineq	A matrix for inequality constraints
Bineq	b vector for results of Ax = b equation
Aeq	A matrix for equality constraints
Beq	B vector for results of Ax = b equation
LB	Lower bound on evaluation variables
UB	Upper bound on evaluation variables
nonlcon	Nonlinear constraint function
randstate	*(Optional)* reset `rand` state for optimization
randnstate	*(Optional)* reset `randn` state for optimization

allows for the user to define specific parameters discussed in section 2.3.2, customized to how the search algorithm should operate. Appendix A gives the MATLAB® GA *options* structure, defining the different options criteria for implementing the algorithm. Further discussion will cover how this research will implement some of those options. While this function is minimized to find the local minimum at $(5, 4)$, the GA toolbox can also find the maximum within a search space. The input function maximum can be found through giving the GA function the negative complement of (5.1), $-Z = -a(x - x_o)^2 - b(y - y_o)$. Various iterations were run for (5.2), ranging from the unbounded case to the constrained case with linear inequalities. For linear constraints, equation (5.12) gives the generic definition for the MATLAB® matrices used in limiting the GA search.

$$\bar{A}\bar{x} \leq \bar{b} \tag{5.12}$$

$$\bar{A} = \begin{bmatrix} a_{(1,1)} & \dots & a_{(1,M)} \\ a_{(2,1)} & \dots & a_{2,2)} \\ \vdots & \ddots & \vdots \\ a_{(N,M-1)} & \dots & a_{(N,M)} \end{bmatrix}_{NxM} \tag{5.13}$$

$$\bar{b} = \begin{bmatrix} b_{(1,1)} \\ \vdots \\ b_{(M,1)} \end{bmatrix}_{M\times 1} \tag{5.14}$$

These equations are defined to enclose the bowl to a smaller space. If matrices (5.13)-(5.14) are listed as `Aeq` and `Beq` instead of `Aineq` and `Bineq`, the GA selects chromosomes from population members on the boundary defined by $\bar{A}$ and $\bar{b}$. The lower bound and upper bound arrays are defined similarly to $\bar{b}$ as a $1 \times N$ array, but bound the variables for the search space:

$$UB = \begin{bmatrix} UB_{(1,1)} & \dots & UB_{(M,1)} \end{bmatrix}^{\dagger}_{M\times 1}, \tag{5.15}$$

$$LB = \begin{bmatrix} LB_{(1,1)} & \dots & LB_{(M,1)} \end{bmatrix}^{\dagger}_{M\times 1}, \tag{5.16}$$

Table 5.2: GA Output Variables

Variable	**Definition**
x	End value for the fitness function
fval	Fitness function value at output x
exitflag	Integer value identifying why the GA ended
output	Output structure that gives performance specifications on the algorithm

where entries must exist for each variable contained in the optimization chromosomes. Any individual gene that does not have an upper or lower bound requires an entry of $\pm\infty$. In the test example, the lower bound was defined as $[-20, -20]$ and the upper bound was defined as $[20, 20]$. This serves an important role in functions that may have more than one minimum or maximum but only a specific one is desired. Once the population space and constraints are defined for $\bar{u}$, the appropriate output variables are explained to ensure the proper results are recorded. The GA output variables produce the actual chromosome $\bar{u}$, the reason why the GA terminated, or the GA scoring value for the chromosome returned. The output values given from running the GA are listed in Table 5.2 in the order that the user can request them. The first two arguments of the GA output give the basic results desired from any minimum/maximum problem solution. The *exitflag* variable gives a scalar number that represents the reason why the GA algorithm terminated. The termination reasons relevant to this research are listed in Table 5.3, which gives explanation as to why the

Table 5.3: MATLAB® GA Exit Flags [37]

1	The average change in fitness function is less than the TolFun and constraint violation is less than TolCon.
3	The fitness function did not change in the generation and the TolCon condition is met.
0	The maximum number of generations specified has been exceed, which the resulting vector would be the smallest in the current population.
-1	The optimization terminated by output or plotting function.
-2	No feasible population starting point was found.
-4	Time limit specified has been exceeded.

Table 5.4: *Output* Structure Variables

Variable	**Definition**
randstate	The state of *rand* when the algorithm started.
randnstate	The state of *randn* when the algorithm started.
generations	The number of generations computed.
funccount	The number of evaluations of the fitness function.
message	Reason that the algorithm terminated.
maxconstraint	Maximum constraint violation, if any occured.

GA terminated and their associated flag values. The TolFun and TolCon conditions, which default at values of 10^{-6}, are defined in the options structure in MATLAB® GA function. The TolFun option defines the average change in a generation that if achieved causes the GA algorithm to exit and the TolCon is the deviation from the linear equality solution allowed for the chromosome to be valid. The last flag (-4) serves as a method, which should be explored in future research, to enable possible real-time GA optimization for engagement scenarios. For current work, this flag is not used because the true threshold is unlimited. Finally, the *output* variable gives a structure with various performance specifications from running the algorithm. Table 5.4 gives the parameters from the *output* structure. The *randstate* and *randnstate* variables allow for the recreation of MATLAB®ś GA results from the seed variables. The *message* and *maxconstraint* variables allow the user to observe how the GA is processing the fitness function. Additional variables for reporting the final population scores and the members in the final population can also be accessed, but were not used.

5.1.3 GA Bowl Optimization Results. The optimization trials were run in four different iterations, covering the different input parameters and limitations for the GA toolbox. Each trial was run in a Monte Carlo simulation of 1000 iterations, with each optimization run from a different random start point in the solution space. The first test setup was run with the unconstrained, unbounded problem. This allowed the GA to search all possible population members to find the problem global minimum. The second trial gave lower and upper bounds of $[-20, -20]$ and $[20, 20]$ respectively

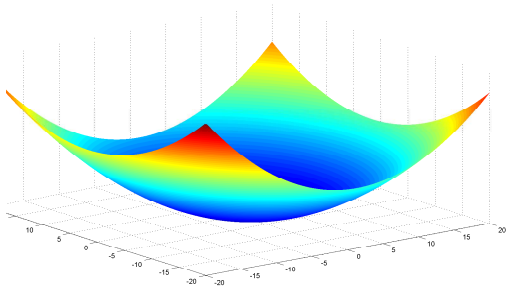


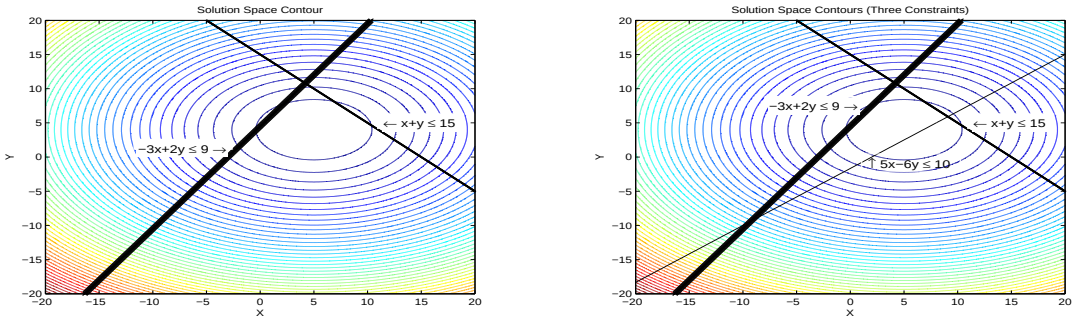
Figure 5.1: Bowl optimization surface from (5.2).

to determine whether or not bounds would decrease solution time or increase accuracy. Figure 5.1 shows Equation (5.2) with the bounds applied. The third trial bounds the search space with the linear inequality constraints defined in (5.17) and (5.18), as:

$$\bar{A} = \begin{bmatrix} 1 & 1 \\ 3 & -2 \end{bmatrix}, \tag{5.17}$$

$$\bar{b} = \begin{bmatrix} 15 \\ 9 \end{bmatrix}. \tag{5.18}$$

During the final simulation, the GA was given an overdetermined system, a system with more equations than unknowns [38]. Having fewer unknowns than defined equations showed how the GA would handle the possibility of conflicting constraints. While the simple bowl problem has a defined solution, the RGPO waveform has different solutions based upon the environmental constraints, which could conflict depending on the desired result. Equations (5.19) and (5.20) give the overdetermined system,



(a) (b)

Figure 5.2: (a) Solution space contour plot with imposed linear inequalities. (b) Solution space with overdetermined linear inequality matrix.

with the last row being inserted.

$$\bar{A} = \begin{bmatrix} 1 & 1 \\ 3 & -2 \\ 5 & -6 \end{bmatrix} \tag{5.19}$$

$$\bar{b} = \begin{bmatrix} 15 \\ 9 \\ 10 \end{bmatrix} \tag{5.20}$$

The final trial run explored GA limitations when given limited information about the optimized function. This trial run mirrors physical constraints placed upon the RGPO waveforms, as discussed in Section 4.2.3. Each linear constraint was defined to contain the center point derived from equation (5.2). Figures 5.2(a) and 5.2(b) give a contour plot of the bowl optimization problem from Section 5.1.1. Figure 5.2(a) shows (5.17) and (5.18) imposed upon the GA optimization. Further constraints are placed upon the GA in Figure 5.2(b), with a third inequality imposed. The third inequality used to make (5.19) and (5.20) had to be chosen such that the desired solution was contained in the enclosed area, shown in Figure 5.2(b). If the third inequality is not carefully chosen, then the overdetermined $\bar{A}$ and $\bar{b}$ matrices can prevent the GA from finding the final solution. Table 5.5 lists the test results after running each limiting

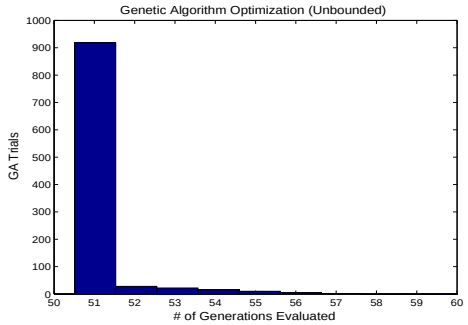
Table 5.5: GA Optimization Statistics

GA Type	$\mu_s(x)$	$\epsilon_x(\%)$	$\sigma_s(x)$	$\mu_s(y)$	$\epsilon_y(\%)$	$\sigma_s(y)$
Unconstrained	5.000	0.00	0.011	4.000	0.00	0.009
Bounded	4.912	1.76	0.045	3.904	2.40	0.050
Linear Constraints	5.001	0.02	0.016	4.000	0.00	0.015
Over-constrained	5.000	0.00	0.012	4.001	0.03	0.013

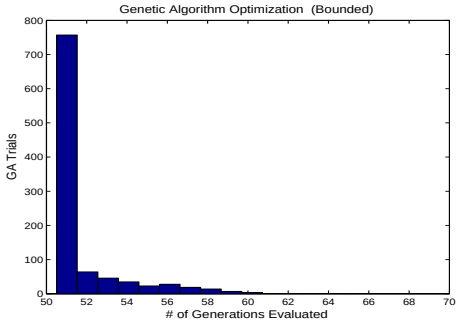
factor with the GA. The statistics given in the table represent the sample mean for each variable, (μ_x, μ_y), the error values from the analytic solution (ϵ_x, ϵ_y), and the sample set standard deviation (σ_x, σ_y). The best results from these trial runs came from the unbounded, unconstrained trials.

During the unconstrained trials, the GA is able to search all values in the variable space without having to determine if generated chromosomes are valid. The linear constraints defined in (5.17) and (5.18) gave the GA constraints to compare chromosomes against to ensure each belonged in the population. Furthermore, the GA could remove invalid chromosomes in the population and had a defined search space to seed the starting generation. The over-constrained trial runs had more error compared to the unconstrained or linear constrained case because the possibility exists where numerous solutions could give the same answer. While the over-constrained result gives relatively poor results, all solution statistics fall within 99% accuracy of the analytic solution. These results translate to measurements off by less than 10kHz on a 1MHz bandwidth signal.

Figures 5.3 and 5.4 show the number of generations required to solve the minimization problem from Section 5.1.1. Each generation contains a population of 100 chromosomes that were evaluated by the fitness function. The histograms in Figure 5.3 show the number of generations needed during each trial of the Monte Carlo simulation to produce the results for Table 5.5. The unbounded, unconstrained simulation took 51-57 generations to produce results, while imposing bounds on (x, y) took longer to find the final result. The final results between these two cases took an average of 20 more generations to derive the solution. The longer evolution time di-



(a) Unconstrained GA generations histogram.

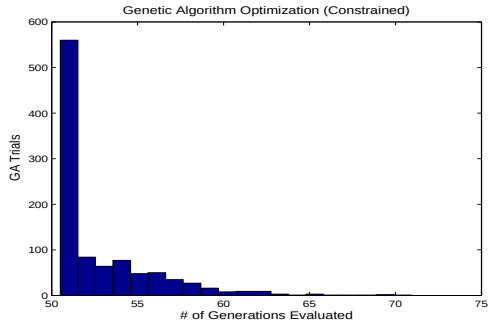


(b) Bounded GA generation histogram.

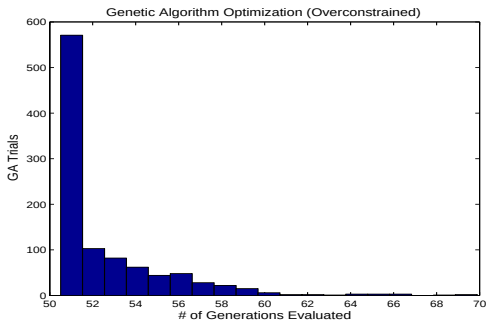
Figure 5.3: GA fitness function histograms.

rectly results from migration towards the minimum happening in smaller increments. In the unbounded case, the large range of fitness values allowed the large values to be discarded more often than the bounded case, which had a smaller range of fitness values, in creating the next generation. The bounded case leads to a smaller search space and a slower descent towards the absolute minimum. The bounded case then leaves more chromosomes in the population that deviate from the solution.

Figure 5.4 shows similar histogram values to the unbounded case. By imposing linear constraints, certain values of the bounded region can be discarded in creating chromosomes during the mutation/crossover stage of the GA. Where the linear constraint problem differs from the unbounded problem is through added constraints to



(a) GA generation histograms with applied linear inequality constraints.



(b) GA generations histogram with overdetermined linear inequality constraints.

Figure 5.4: GA results from applying linear inequality constraints.

limit the population, which equates to the histogram's standard deviation. Figure 5.4(a) contains values that range from 50-70 generations, 10% more than the 10 generation span of Figure 5.3(a). Again, this variation comes from the discussed issue of retained chromosomes. Finally, Figure 5.4(b) contains the same logarithmic shape as the three previous plots. Careful selection of the third linear inequality clustered more of the generation results to the left half of the histogram. These results combined with Table 5.5 shows that, although more linear inequality equations than unknowns can harm the final solution, judicious selection of constrains can lead to refined results.

5.1.4 Performing a Linear Transformation on the Fitness Function. The next exploration area for implementing GA optimization was performing a linear transform on the observation space. Performing a linear transform on the GA fitness function represents effects caused by the radar environment on the received RGPO waveform. In order to understand how the GA will handle transformations of the initial optimized function, a simple linear transformation matrix was developed to manipulate the previously optimized bowl function. Figure 5.5 shows the population undergoing a linear transform $\mathcal{L}$, as shown in the HILS architecture, implemented by the L-matrix (L):

$$L = \begin{bmatrix} 2 & 4 \\ 0 & -2 \end{bmatrix}, \tag{5.21}$$

which performs two different operations on $\Phi(\bar{u})$. First, the fitness function is reflected on the x-axis. The fitness function then has a stretch factor of 2 applied to the solution space before a vertical sheer mapping factor of 4 applied before the result depicted in Figure 5.5 [38]. The analytic solution for the linear transformation problem then comes from applying the L-matrix to the solution space. Equation (5.22) shows the algebraic representation of the analytic solution matrix $\bar{z}$ transformed by the L-matrix, giving the result as $\bar{z}$:

$$\bar{z} = \bar{L}^{-1}\bar{c} \tag{5.22}$$

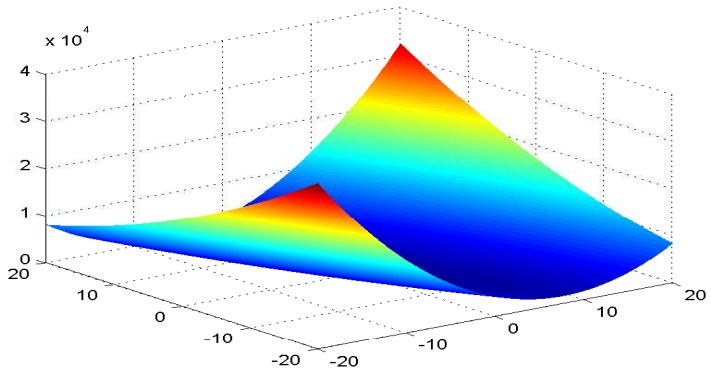


Figure 5.5: Linear transformation of bowl optimization problem.

where

$$\bar{L}^{-1} = -\frac{1}{4}\begin{bmatrix} -2 & -4 \\ 0 & 2 \end{bmatrix} \tag{5.23}$$

$$= \begin{bmatrix} .5 & 1 \\ 0 & -.5 \end{bmatrix}. \tag{5.24}$$

This transformation on the solution produces

$$\bar{z} = \begin{bmatrix} .5 & 1 \\ 0 & -.5 \end{bmatrix}\begin{bmatrix} 5 \\ 4 \end{bmatrix} \tag{5.25}$$

$$= \begin{bmatrix} 6.5 \\ -2 \end{bmatrix}, \tag{5.26}$$

which the results can be verified visually in Figure 5.5. The simulated linear transform represents the necessity for the GA to wait for the radar signal processor to detect the ECM signal and develop a track profile on the false target before returning a result to the system.

The test setup for the applied linear transform is similar to the test scenario for solving the bowl problem. Each setup was run under the same number of trial runs and random initialization points to determine the results shown in Table 5.6.

Table 5.6: Linear Transformation Statistics

GA Type	$\mu_s(x)$	$\epsilon_x(\%)$	$\sigma_s(x)$	$\mu_s(y)$	$\epsilon_y(\%)$	$\sigma_s(y)$
Unconstrained	6.497	0.04	0.048	-1.999	0.05	0.022
Bounded	6.497	0.04	0.056	-1.999	0.05	0.026
Linear Constraints	6.492	0.12	0.059	-1.996	0.02	0.026
Over-constrained	6.489	0.18	0.062	-1.995	0.03	0.030

Table 5.6 contains the all the same variables and significant statistics as illustrated earlier in Table 5.5. There are a few important results that come from the results shown here in Table 5.6. The first is that (ϵ_x, ϵ_y) for the unconstrained, bounded, and linear constraints test all fall within 99.00% as the previous tests did. The linear transformation applied to the problem space did not change how accurate the GA was in finding the minimum solution. The linear inequality matrix, $\bar{A}_i$, and vector, $\bar{b}$, were selected using equations chosen that incorporated the translated new solution. Both $\bar{A}$ and $\bar{b}$ chosen for the linear transformation GA simulations are shown in equations (5.27) and (5.29):

$$\bar{A}_i = \begin{bmatrix} .5 & .5 \\ 1.5 & 4 \end{bmatrix} = \bar{A}\bar{L}^{-1} \tag{5.27}$$

$$\bar{b}_i = \bar{A}\bar{z} = \bar{A}\bar{L}^{-1}\bar{c} \tag{5.28}$$

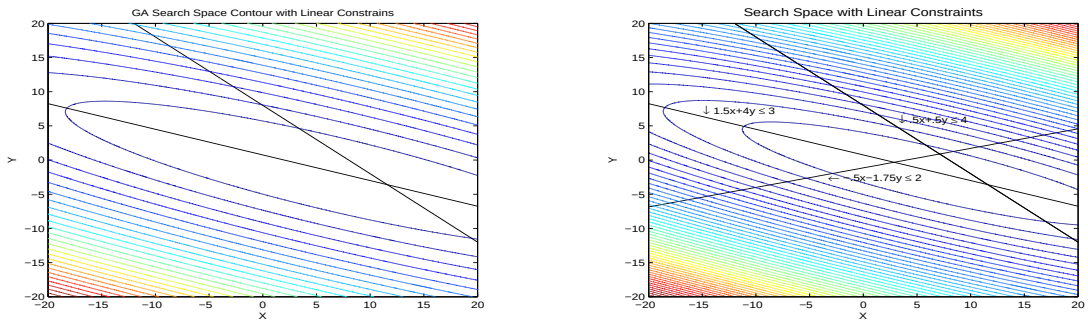
$$= \begin{bmatrix} 2.25 \\ 1.75 \end{bmatrix}. \tag{5.29}$$

The overdetermined problem contained an additional linear inequality added, as given in (5.30)-(5.32):

$$\bar{A}_i = \begin{bmatrix} .5 & .5 \\ 1.5 & 4 \\ .5 & -1.75 \end{bmatrix} = \bar{A}\bar{L}^{-1} \tag{5.30}$$

$$\bar{b}_i = \bar{A}\bar{z} = \bar{A}\bar{L}^{-1}\bar{c} \tag{5.31}$$

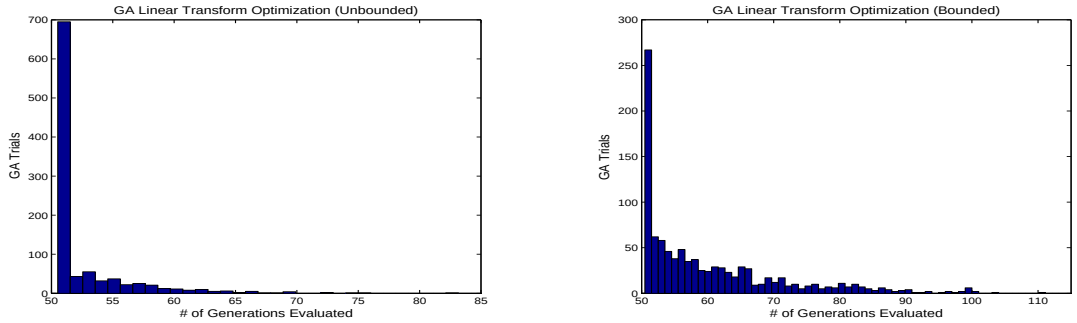
$$= \begin{bmatrix} 4 & 3 & -2 \end{bmatrix}^{\dagger}. \tag{5.32}$$



(a) (b)

Figure 5.6: (a) Solution space contour plot with imposed linear inequalities. (b) Solution space with overdetermined linear inequality matrix.

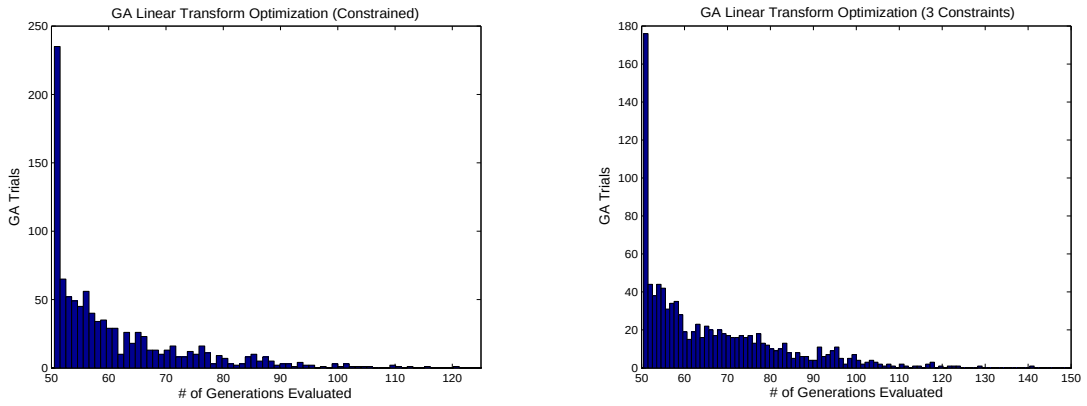
The arrays shown in (5.30)-(5.32) are to expand the overdetermined problem beyond the solved solution. Through expansion of this search space, the GA is constrained to a search area, but that area does not collapse on one individual point. Looking back at Figure 5.5 shows that the bottom of the transformed bowl has flattened out, leaving a smaller gradient change between the minimum, $[6.5, -2]$ and its surrounding points. Figures 5.6(a) and 5.6(b) give the transformed bowl contour plot, depicting the flattened bottom. These two figures depict the constraints imposed from the equations (5.27)-(5.32), attempting to impose physical limitations to the defined linear transform. The $(\sigma_s(x), \sigma_s(y))$ in Table 5.6 show the results of the bowl's flattened bottom because these values are significantly greater than the original optimization problem.



(a) GA generations histogram with no bounds (b) GA generations histogram with upper and lower bounds applied

Figure 5.7: Linear transform applied to GA fitness function

The histograms in Figure 5.7 show the number of generations needed during each trial of the Monte Carlo simulation to produce the results for Table 5.6. The unbounded results, shown in Figure 5.7(a), have similar results to the original unbounded case. As with the untransformed GA results, more than 80% of the unconstrained GA trials resolved a solution within 50-55 generations. Promising results from this simulation come from knowing that the maximum number of generations necessary to resolve the solution in the bounded case was 80. The bounded case shown in Figure 5.7(b) also shows similar results to the unbounded case, but a less defined peak of 50 generations needed to develop the final solution. The linear transformed bounded case resolved in under 65 iterations for most trials, but 15% of all trials took more than 70 iterations to resolve. Figure 5.8 shows the results to the constrained cases of the applied linear transform on equation (5.2). In the linear inequality histogram, shown in Figure 5.8(a), fewer trials finished in the peak of 50 generations, but the majority of the trials fall within the 50-70 generations range. This is consistent with the results from the unbounded case, but does have outliers falling as far away as 120 generations. These outliers are farther spread from the mean number of generations needed, which is consistent with having a transform applied to the fitness function.



(a) GA generations histogram with linear inequalities applied.

(b) GA generations histogram with overdetermined linear inequality applied.

Figure 5.8: Linear transform applied to constrained GA fitness function with applied linear inequalities.

The convergence rate is slower in the linear transformed fitness function case as seen in Figure 5.8(a) due to the spreading of the bowl's bottom. This same reasoning explains the histogram spreading for the overdetermined linear inequality simulation shown in Figure 5.8(b).

The number of generations for Figure 5.8(b) continue the trend illustrated in the previous three histograms. This histogram's shape mirrors the other simulations with the L^{-1} transform applied, such that all four have an exponential decay pattern. Only 176 trials completed in 51 generations, which continues to show the solution space having a gradual slope. The spreading of histogram values for Figure 5.8(b) shows the GAs ability to resolve the solution over time. The 140 generations maximum for the GA to resolve the solution modeled in Figure 5.8(b) would cause concern in modeling mathematical functions. This result shows that the landscape definition can be expressed concisely through mathematical equations and still take numerous generations because of gradual changes.

The modeling of a linear transformation on a simple bowl problem illuminates the point of judiciously developing the mathematical representation of the physical limitations on the RGPO and VGPO waveforms. Understanding the landscape of the ECM library functions allows precise GA implementation. Conversely, if the ECCM limitations and the physical environment are not carefully described, the GA would produce erroneous parameters that may not work against the threat radar system. The next section covers how the MATLAB® GA functions optimized the defined mathematical functions for the range gate pull-off waveform.

5.2 GA RGPO Implementation

For optimization of the RGPO signal, certain physical environmental limitations must be accurately modelled. The optimization parameters are defined from Section 4.2.1, expressed as:

$$u = \begin{bmatrix} R_o & R_{\max} & T_w & f & JSR \end{bmatrix}^{\dagger} \tag{5.33}$$

The optimization routine without any upper or lower bounds serves no meaning to developing accurate deception signals. Each parameter must be given a range of values to search from to the domain of the values, which is defined in Table 5.7. The polynomial factor for the RGPO profile maximum is the linear acceleration case due to physical modeling constraints. Although section 4.2.1.4 discusses the possibility of higher-order profile, current hardware limitations are contained to the linear acceleration model. The JSR domain is also limited in scope, based upon the jammer hardware modelled in the HILS architecture. This domain limitation becomes necessary to prevent RGPO waveforms exceeding the known hardware capabilities.

Further constraints are placed upon the optimization parameters through (5.12) as implemented in the bowl optimization problem. These constraints take on physical limitations relative to the radar environment or the engagement scenario. Equation (5.34) and (5.35) gives an example of the necessary constraints for the MATLAB® GA algorithm, where $\bar{A_e}$ and $\bar{b_e}$ represent the matrix and vector components respectively.

$$\bar{A_e} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{5.34}$$

$$\bar{b_e} = \begin{bmatrix} 3 & G_{j_{\max}} & 0 \end{bmatrix}^{\dagger} \tag{5.35}$$

$\bar{A_e}$ and $\bar{b_e}$ show some of the necessary limitations on the parameters shown in Table 5.7. The first row limits the f-nomial RGPO function to the linear acceleration case, ensuring that only the three specific cases covered in Section 4.2.1. The second row

Table 5.7: RGPO Optimization Parameter Domain

Parameter	**Domain**
R_o	$[0, \infty)$
$R_{\max}$	$[0, \infty)$
T_w	$[0, \infty)$
f	$[0, 3]$
JSR	$[0, G_{j_{\max}})$

states that the maximum JSR possible must be within the limits of specified jammer. For this research, the maximum jammer signal equates to $10dBm$ and the maximum jammer gain is stated as $30dB$ [23]. The last row given in this set of equations states that $R_{\max}$ should be larger than R_o, as defined in the RGPO profile. The first subsection describes the environmental considerations factored in when implementing the GA with the HILS architecture. The scoring function for this architecture will be discussed further during the first subsection, based upon how the scoring processes the environmental considerations. The second subsection describes the results derived from implementing specific $\bar{A}_r$ and $\bar{b_r}$ for optimization.

5.2.1 HILS Architecture Considerations for GA Implementation. While $\bar{A}_e$ and $\bar{b_e}$ serve as possible linear constraints on the function, more explanation and better development must be discussed to ensure proper GA implementation. The first consideration that requires implementation is that $R_{\max}$ should be larger than R_o to ensure that the RGPO signal integrity stays intact. If $R_{\max} = R_o$, the profile becomes a false target generator, which is undesired for this implementation. The first line of (5.36) and (5.37) ensures this condition in $\bar{A}_r$ and $\bar{b_r}$, which are expressed in 5.37 as the MATLAB® linear inequality.

$$\bar{A}_r = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -PRF \end{bmatrix} \tag{5.36}$$

$$\bar{b_r} = \begin{bmatrix} 0 & -R_{res} & -3R_{res} & 6 & 10 & -2 \end{bmatrix}^{\dagger} \tag{5.37}$$

The second and third lines of $\bar{A}_r$ and $\bar{b_r}$ state that both R_o and $R_{\max}$ are to be larger than the range resolution and three times the range resolution respectively. These limitations set a minimum walk-off range of at least two pulse widths. The fourth

row represents that the maximum walk-off time is six seconds for this engagement. Row five expresses that the maximum pull-off range cannot exceed 10m, which is beyond the Lab-Volt™ś detection range [4]. The last row imposes the limitation that the walk-off time must span at least two PRIs. This inequality prevents the GA from generating false targets to inject into the environment. The matrix $barA_r$ and vector $\bar{b}_r$ ensure the maximum walk-off length, that the pulse train walks the target away in distance instead of closes the distance, and ensures that the power profile decreases. No specific limitations are placed upon the polynomial factor. The polynomial factor is already limited to integer values given in Table 5.7, which only needs further limitation if the specific profile were to be optimized on. The GA implementation has been set to explore the different values of f.

The other necessary limitation implemented were nonlinear constraints, to impose velocity and acceleration constraints. In the GA input options, a MATLAB® script can be defined to impose these limitations in the following general form:

$$\bar{c}_{\text{eq}} \leq 0, \tag{5.38}$$

$$\bar{c} = 0. \tag{5.39}$$

Equation (5.38) represents the interaction between variables that specify certain limits while (5.39) represents boundary conditions. Special care is taken to define $\bar{c}$ and $\bar{c}_{\text{eq}}$ to prevent errors. If these parameters are not adequately defined, the GA does not find a desirable initial population and exits prematurely. Equation (5.40) represent the limitations imposed based upon the Lab-Volt™ simulation:

$$\bar{c}_{\text{eq}} = \left[\begin{array}{ccc} -\frac{R_{\max}-R_o}{T_w} + .15 & -\frac{R_{\max}-R_o}{T_w} + -1 & -\frac{R_{\max}-R_o}{{T_w}^2} - (3 * 9.8) \end{array} \right] \tag{5.40}$$

The first two lines of (5.40) give a minimum velocity of $.15\frac{\text{m}}{\text{s}}$ and maximum velocity of $1.00\frac{\text{m}}{\text{s}}$, which are relative to the motion of the Lab-Volt™ target table [29]. The final line of (5.40) defines the maximum acceleration as three times gravity. The max-

imum acceleration mirrors movement of a realistic target, stated in Section 4.2.3. A limitation to using nonlinear constraints is the ability to use the customized mutation functions in the GA toolbox. The **MutationFcn** setting, shown in Appendix A, must be set to `@mutationadaptfeasible` due to population generation. Although this setting was not explored during this research, future work should consider developing mutation functions to increase search efficiency.

The scoring function developed for the GA implementation mirrors the discussion from section 3.2.5. After receiving the processed Simulink data, the detector determines the power level where the true target should return. This power level is set as the minimum value needed by the RGPO signal to deceive the radar receiver. This range bin is set as the default value in searching the remaining PRIs for the new target. The MATLAB® scoring function then determines where the power level of the target return exceeds the false target power level and marks that PRI in the sequence as w. The scoring value for all chromosomes is defined as:

$$\mathcal{K}[\mathcal{L}[\Phi(u)]] = -\left(\frac{w}{W}\right). \tag{5.41}$$

$\mathcal{K}[\mathcal{L}[\Phi(u)]]$ has been negated because the MATLAB® GA toolbox finds the expressed function minimum, as discussed in Section 5.1. This function operates independent of JSR, because the power level is integrated into the Simulink model and w is based upon detection criteria determined within the model. While no ECCM has been set for automatic gain control, the equation weights the result to produce the desired walk-off of the false target and meet the power profile necessary. Another benefit to this scoring function is that profiles that are too long but partially deceive the radar system are kept in the sample set. Those chromosomes meeting this criteria are ranked higher in the population and maintain a high probability of reproducing into following generations.

Finally, adjustments were made to the GA optimization settings to allow for search space exploration. The first setting changed was running the GA 100 times for the GA Monte Carlo simulation. The GA optimization could either collapse upon a specific set of radar parameters, as shown in the initial optimization problem, or could derive different solutions based upon where the GA started. A Monte Carlo simulation was run to average out the possibility of a specific instance finding local minima. Although previous section discussed running 1000 iterations for the Monte Carlo simulation, the HILS architecture run with the GA consumes more time than necessary to sample the produced results. Conversely, a few iterations would not give enough information about the RGPO waveform landscape. The population size was reduced to 20 members, which is the default setting for the MATLAB® GA. Trial runs of the HILS architecture showed that large population sizes took significantly longer than was required. Furthermore, the nonlinear constraints increased the landscape search, as nonconforming chromosomes were rejected based upon the *TolCon* defined in Appendix A. Another parameter changed involved increasing the *StallTimeLimit* to infinity. This prevented the GA from prematurely ending if the Simulink models took too long to process each generated population. The last parameter changed was using the tournament selection function for the *SelectionFcn* option. This rates each chromosome from the scoring value and determines which chromosomes should be kept for reproduction. By ordering the results in this manner, the undesired results discussed earlier would drop out and be replaced by viable chromosomes.

5.2.2 RGPO Optimization Results. The HILS architecture optimization subspace for the RGPO waveform has five unique dimensions which requires careful analysis to understand the results. Figure 5.9 shows an example walk-off profile from the Monte Carlo simulation for the GA optimization and its defined amplitude and range profiles. Figure 5.10 shows the matching range delay and amplitude value for each pulse in the GA's optimized solution. These figures are created from the

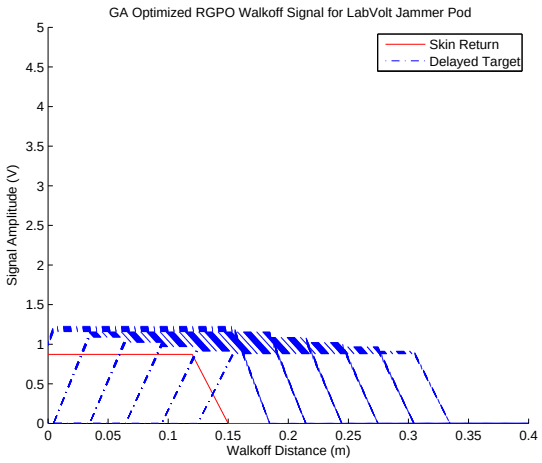
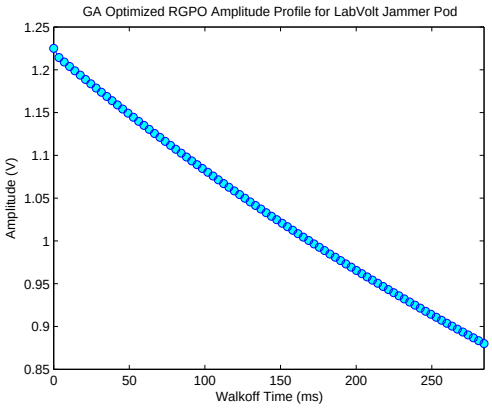


Figure 5.9: GA optimized RGPO walk-off signal for Lab-Volt™ Jammer.

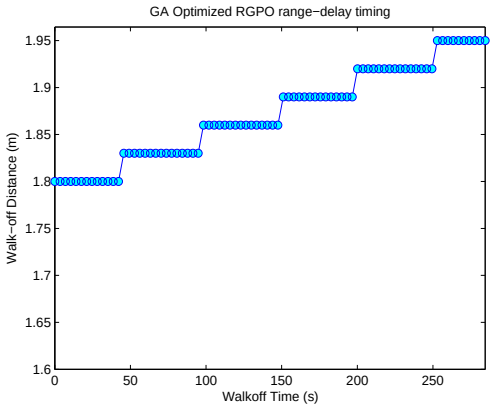
chromosome:

$$\bar{u}_c = \begin{bmatrix} 0.0046 & 0.1645 & 0.2845 & 1 & 3.002 \end{bmatrix}, \tag{5.42}$$

which is the result from a single GA optimization trial within the HILS architecture. The amplitude profile of Figure 5.10(a) shows the decay as expected from the discussion in Section 4.2.2. The walk-off distance shown in Figure 5.10(b) has discrete steps based upon the radar model's range resolution. A comparison of Figure 4.12 to the GA optimized solution shown in Figure 5.9 have walk-off profiles with a finite number of pulses displayed. The Lab-Volt™ system parameters set $N = 8$,

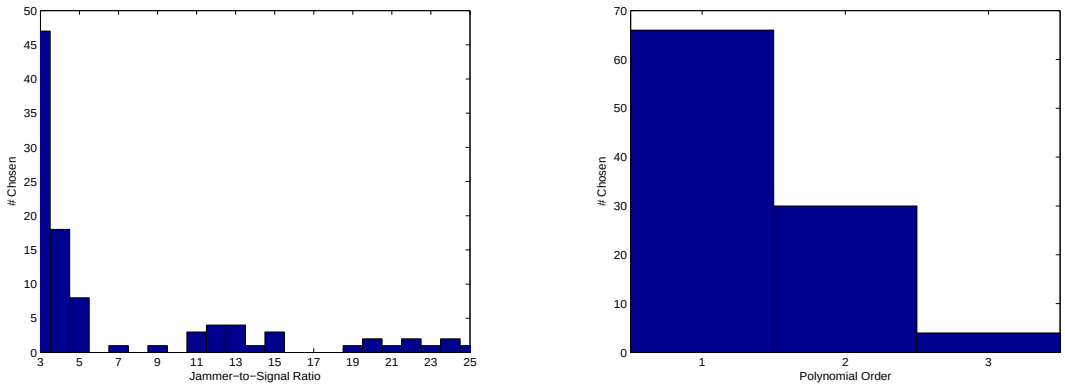


(a) Optimized RGPO Amplitude Profile.



(b) Optimized RGPO Range-Delay Profile.

Figure 5.10: Optimized Lab-Volt™ RGPO range and amplitude profiles.



(a) GA selection of Jammer-to-Signal ratio.

(b) GA selection for polynomial order.

Figure 5.11: HILS architecture Monte Carlo simulation histograms.

independent of the desired walk-off time. Figure 5.10 shows 6 distinct pulse delays shown in the profile, although using Equation (4.20) results in 82 unique pulse delays. The Lab-Volt™ system determines the number of pulses per delay step required to walk-off the false target. This optimization solution contains 15 pulses before the walk-off distance is incremented. The false target velocity for $\bar{u}_c$ is $0.56\frac{\text{m}}{\text{s}}$, which is close to the Lab-Volt™ system's calculated false target velocity of $0.64\frac{\text{m}}{\text{s}}$ for the 0.8s walk-off time. The GA optimized solution shows the modelled RGPO waveforms for the Lab-Volt™ Jammer Pod may not be optimum for the purpose for demonstration. The total walk-off distance is shorter for the optimization solution, but also uses a smaller R_o prior to pulling off the target.

The GA optimization routine produced $\bar{u}_c$ in 18 minutes, which is significantly shorter than it would take a man-in-the-loop system to produce. Although the MATLAB® GA operates with continuous variables, an exhaustive search of this same 5-D landscape takes orders of magnitude larger to produce a similar result. If each variable were divided into discrete search spaces of 100 elements, the exhaustive search could take 100^5 or 100 million iterations. With nonlinear constraints rejecting population members prior to evaluation, the GA evaluates only desired parameter values and subsequently reducing the number of iterations by at least an order of magni-

tude. All 100 GA optimization simulations took less than 10 generations to evaluate for the elite solution. The Simulink model initialization consisted of the majority of the solution time. Further research is necessary to ensure simulation efficiency.

Table 5.8 gives aggregate statistics from running the HILS Architecture Monte Carlo simulation. The raw data is contained in Appendix B formatted in accordance with Equation 5.33. The average maximum velocity from these statistics is $0.353\frac{\text{m}}{\text{s}}$, which closely resembles the Lab-Volt™ Jammer Pod false target velocity of $0.32\frac{\text{m}}{\text{s}}$ for its walk-off time of 1.6s, as stated in Section 4.2. Figure 5.11(a) shows the histogram of JNS values based upon the GA optimization. The JNS values range between $3-25$ dB, with 64.0% fall between $3-8$dB. With no ECCM modelled in the HILS architecture, the chromosome scores -1 for any profile that achieves the pull-off with an amplitude much larger than the true target. Further implementation should explore using limits in the automatic gain control to determine RGPO profiles. Figure 5.11(b) shows the Monte Carlo results for the specific polynomial profile chosen. The constant velocity profile was chosen 66% of the time, as expected from the Lab-Volt™ system simulation discussion from Section 4.2.4. Only 4% of the simulations selected the linear acceleration model, which would be the least likely option for modeling the Lab-Volt™ deception signal. The range of values for $R_{\max}$ covering 1m does not allow linear acceleration motion changes to be apparent in the RGPO profile. The linear velocity profile ($f = 2$) was selected 30% of the time, which suggests that this profile reasonably walks off the true target. Although the constant velocity profile solution is accurate, results also suggest that certain walk-off distances could use a

Table 5.8: GA Monte Carlo Simulation Results.

Variable	μ_c	σ_c
R_o (m)	0.2131	0.1729
$R_{\max}$ (m)	0.6816	0.4947
T_w (s)	1.3266	1.3482
f	1	0.6567
JNS (dB)	6.8344	4.6534

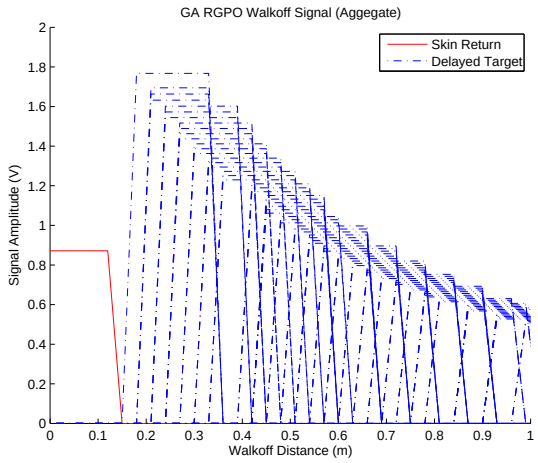
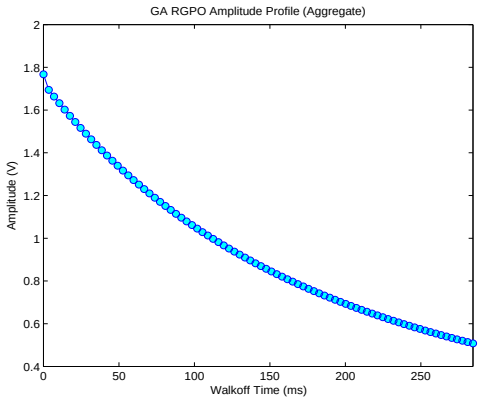


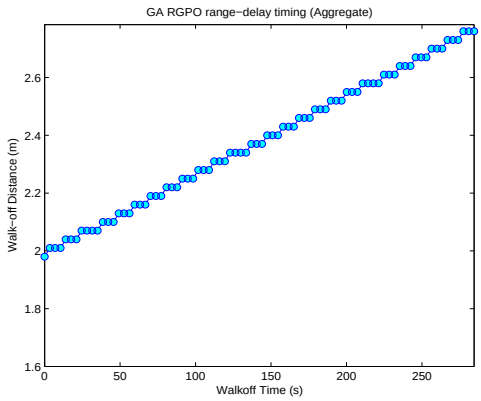
Figure 5.12: Aggregate GA RGPO walk-off profile for Lab-Volt™ Jammer.

more-realistic linear velocity model with accuracy to pull-off the Lab-Volt™ target tracker.

Figure 5.12 shows the aggregate RGPO walk-off profile for the Monte Carlo simulation, using the statistics from Table 5.8. The walk-off distance and R_o are close to the Lab-Volt™ Jammer Pod RGPO parameters given in [23]. The Jammer-to-Signal ratio from Table 5.8 does not produce a power level large enough to deceive the radar through the entire profile. The possibility of multiple walk-off distances with multiple JNS ratios within the landscape prevents direct correlation between the aggregate JNS ratio and the desired wall-off profile. Under the HILS architecture scoring system, the Monte Carlo simulation results in a fitness value of 0.518. Although the individual solutions resulted in a fitness value of 1.0, the aggregate shows the boundary conditions are not defined accurately enough to determine a singular chromosome to model the RGPO waveform. Table 5.9 collects the Monte Carlo simulation data based upon Jammer-to-Signal ratios, as shown from Figure 5.11(a). As shown from the scoring function for each aggregate result, the $3 - 10$ dB solution would not suffice for the given boundary conditions although each individual result is considered an optimum solution. All three averaged solutions meet the nonlinear boundary conditions and the linear boundary conditions discussed in Section 5.2.1.



(a) Aggregate RGPO Amplitude Profile.



(b) Aggregate RGPO Range-Delay Profile.

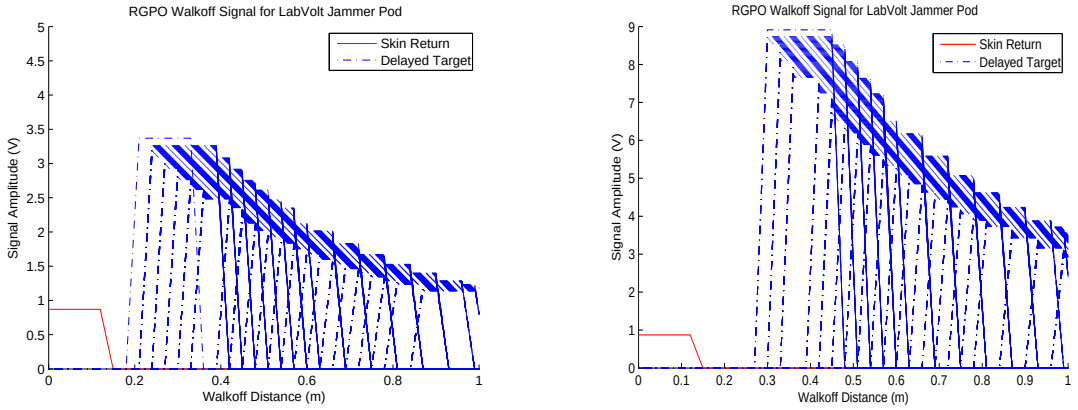
Figure 5.13: GA RGPO profile from Monte Carlo simulation.

Further analysis shows that 75% of the data does not meet the desired result based upon these aggregated results. The two other JNS categories in Table 5.9 do produce a valid optimized result, as shown in Figure 5.14. Comparison of these two profiles shows that the profile shown in Figure 5.14(b) would be rejected if certain ECCM flags were set to reject extreme signal increases. Figure 5.14(a) does not have a drastic signal increase and may deceive automatic gain control levels. Figure 5.14 illustrates that depending upon specific constraints, certain local minimum would serve as the extreme minimum for the solution set. These results show that multiple local minimum exist within this search space based upon the liberal boundary conditions set.

5.2.3 GA Optimization Summary. The MATLAB® GA optimized the HILS architecture using the developed Lab-Volt™ Simulink model. The optimization values show that numerous different RGPO profiles exist based upon the broad limitations

Table 5.9: GA Monte Carlo Results by JNS.

JNS Range	R_o	$R_{\max}$	T_w	JNS	$V_{\max}$	$\mathcal{K}[\mathcal{L}[\Phi(u)]]$
$3-10$	0.1969	0.8563	1.5523	3.6233	0.42	0.12
$10-20$	0.2399	0.6604	1.4865	13.9223	0.28	1.00
$20-25$	0.3168	0.8783	1.6205	23.0131	0.35	1.00



(a) Monte-Carlo Simulation RGPO Profile (JNS=13.92 dB).

(b) Monte-Carlo Simulation RGPO Profile (JNS=23.03 dB).

Figure 5.14: RGPO signal comparison for different JNS ratios.

given in Section 5.2.1. All solutions take on the desired velocity and acceleration limitations as specified by the nonlinear constraints. These results show that the optimization architecture successfully determines RGPO waveforms from the radar models placed in the system. The majority of the RGPO waveforms developed from the Monte Carlo simulation depict similar general characteristics of the Lab-Volt™ Jammer Pod but do not necessary represent a specific extreme minimum. Judicious selection of boundary conditions becomes necessary to ensure that the GA optimized solution produced the global minimum for the specific asset. Careful selection of linear and nonlinear constraints, along with specific boundaries, causes the GA to converge upon the optimum solution.

VI. Conclusions and Future Research

Developing timely ECM waveforms through a HILS implementation with an integrated optimization technique allows for the US Air Force to maintain a favorable advantage in the electromagnetic spectrum as radar systems become increasingly complex. This research explored two major aspects in developing this architecture: Creating the genesis of an ECM Library containing mathematical representations of well-known jamming signals and implementing a genetic algorithm independent of the radar operating environment to optimize the jammer signals. Each area's degree of success is explored in the first two sections. The final section explores the future work that should be explored within the HILS architecture.

6.1 ECM Library Development

The RGPO mathematical modeling was done for the generic polynomial case. The VGPO mathematical modelling was done in a similar fashion in STAP framework. The power profile for both signals was completed with consideration to the one-way transmission equation. The power dissipation for the RGPO and VGPO signals depicts the desired changes to move the false target away from the true target. The option currently exists for the operator to manually select a specific waveform to optimize on and determine its optimum solution. Currently, there is no switching mechanism to select from RGPO to VGPO waveform optimization or to optimize based on both waveforms. Future work should focus on the development of other waveforms, such as the coordinated RGPO/VGPO profile. The signal coordination should be completed using STAP implementation for producing the entire profile at a certain instant. Another area for future efforts focuses on expanding the ECM library functions. A multitude of mathematical functions for the ECM library allows for developing the best jamming technique against a specific asset. In expanding the ECM library, a method should be developed to compare between the different library functions. This research track paves the way of optimizing numerous waveforms at a given instant for the ideal waveform/technique against a particular asset.

6.1.1 RGPO Waveform Modelling. The RGPO modeling functions covered the known motion cases of constant velocity, linear velocity, and linear acceleration. Furthermore, a generic RGPO profile was successfully developed. The generic case showed the signal dependency on three important variables: R_o, $R_{\max}$, and T_w. The waveform model was compared to the Lab-Volt™ jammer specifications and showed how the general case applied to a realistic system. Future research should look to expand upon the generic profile and explore how this profile affects the power necessary to deceive the tracking radar. This exploration should also consider waveform modelling with targets changing profiles, for example from a constant velocity to a linear acceleration, to mimic targets following realistic flight paths.

6.1.2 VGPO Waveform Modelling. The VGPO waveform development was conducted in a STAP framework for implementing the necessary Doppler phase shifts. The VGPO signal model naturally developed from the RGPO general signal model. The STAP framework is not new to radar signal processing but provides a natural format for storing the necessary Doppler shifts in memory for implementation. Proper STAP configuration for the jammer pod allows signal memory to contain the full velocity profile based upon the system platform's own velocity and position. Future research should implement a VGPO jammer pod for use with the Lab-Volt™ system. This work would develop a training tool for operators to understand the VGPO waveform, along with studying how the Lab-Volt™ Target Tracking modules handles velocity gating.

6.2 Optimization Algorithm

The MATLAB® genetic algorithm served as proof of concept for optimization of the RGPO/VGPO waveforms. The genetic algorithm produced significant results from desired boundaries and system constraints. Careful design of the linear inequality constraints produces specific range profile development based upon known physical limitations. The genetic algorithm implementation in MATLAB® can be

time-consuming for each run, but easily integrates with the testing hardware contained in the RAIL research laboratory. Future work should explore the possibility of other optimization routines for use in the HILS configuration. Literature reviewed suggested other methods, such as a direct search or simulated annealing, to explore the waveform development landscape. Results show that the ECM optimization landscape in not straight-forward, requiring more analysis of how each parameter interacts with physical constraints. Research is necessary to compare the various optimization techniques as well to determine which one will best utilize the known search space behavior. Further exploration should also look into the best software optimization implementation. MATLAB® serves as an essential electrical engineering tool, but does not always utilize memory efficiently for this optimization. Implementation in either C++, FORTRAN, or another programming language may produce timely results towards near-real time integration in future systems.

6.3 HILS Architecture

The basic setup for the proposed HILS architecture was exhibited using the MATLAB® genetic algorithm with the developed Simulink model. The individual components are modular such that other software models or physical assets may be substituted for a desired configuration. The scoring function developed conducts rudimentary tracking of the RGPO signal. The scoring function outputs scalar values based upon certain known conditions in the RGPO jammer signal. A higher-fidelity model of the Lab-Volt™ tracking radar is required to ensure that the scoring function accurately portrays the Lab-Volt™ system response. Future work should explore expanding the fidelity of the HILS architecture. Currently, the architecture exists only in software, through MATLAB® and Simulink models. Research efforts should explore implementation outside these environments. Necessary work follows the from the possibility of running the Textronix Lab Equipment in conjunction with the Lab-Volt™ system.

Appendix A. MATLAB® Genetic Algorithm Options

This appendix gives the options structure table used to define the operation parameters of the MATLAB® Genetic Algorithm Toolbox. Defined in the table below are the specific options, their descriptions, the range of values each can take on. The values used during RGPO optimization are italicized and those values that are the MATLAB® GA defaults but not used in this simulation are underlined, unless only one value given.

Table A.1: MATLAB® GA Options Structure [37]

Option	**Description**	**Values**
`CreationFcn`	Handle to the function that creates the initial population	@gacreationuniform
`CrossoverFcn`	Handle to the function that the algorithm uses to create crossover children	*@crossoverscattered* @crossoverintermediate @crossoversinglepoint @crossovertwopoint @crossoverarithmetic
`CrossoverFraction`	The fraction of the population at the next generation, not including elite children, that is created by the crossover function	(0,1) *0.8*
`Display`	Level of display	'off' 'iter' *'final'* 'diagnose'
`EliteCount`	Positive integer specifying how many individuals in the current generation are guaranteed to survive to the next generation	Positive integer: *2*

Table A.1 – Continued

Option	**Description**	**Values**
`FitnessLimit`	Scalar. If the fitness function attains the value of `FitnessLimit`, the algorithm halts.	Scalar - *Inf*
`FitnessScalingFcn`	Handle to the function that scales the values of the fitness function	@fitscalingshiftlinear @fitscalingprop *@fitscalingrank* @fitscalingtop
`Generations`	Positive integer specifying the maximum number of iterations before the algorithm halts	Positive integer | <u>100</u>, *2000*
`HybridFcn`	Handle to a function that continues the optimization after GA terminates	User-Defined Function @fminsearch @patternsearch @fminunc @fmincon *[]*
`InitialPenalty`	Initial value of penalty parameter	Positive scalar | 10
`InitialPopulation`	Initial population used to seed the genetic algorithm	Matrix | []
`InitialScores`	Initial scores used to determine fitness	Column vector | []
`MigrationDirection`	Direction of migration	*'forward'* | 'both'
`MigrationFraction`	Scalar between 0 and 1 specifying the fraction of individuals in each subpopulation that migrates to a different subpopulation	Scalar | *0.2*

Table A.1 – Continued

Option	**Description**	**Values**
`MigrationInterval`	Positive integer specifying the number of generations that take place between migrations of individuals between subpopulations	Positive integer | *20*
`MutationFcn`	Handle to the function that produces mutation children	@mutationuniform @mutationadaptfeasible *@mutationgaussian*
`OutputFcns`	Functions that ga calls at each iteration	@gaoutputgen | *[]*
`PenaltyFactor`	Penalty update parameter	Positive scalar | *100*
`PlotFcns`	Array of handles to functions that plot data computed by the algorithm	@gaplotbestf @gaplotbestindiv @gaplotdistance @gaplotexpectation @gaplotgeneology @gaplotselection @gaplotrange @gaplotscorediversity @gaplotscores @gaplotstopping []
`PlotInterval`	Positive integer specifying the number of generations between consecutive calls to the plot functions	Positive integer | 1

Table A.1 – Continued

Option	**Description**	**Values**
`PopInitRange`	Matrix or vector specifying the range of the individuals in the initial population	Matrix or vector | [0;1]
`PopulationSize`	Size of the population	Positive integer | 20
`PopulationType`	String describing the data type of the population **Note: linear and nonlinear constraints are not satisfied when** `PopulationType` **is set to 'bitString' or 'custom'.**	'bitstring' 'custom' `doubleVector`
`SelectionFcn`	Handle to the function that selects parents of crossover and mutation children	@selectionremainder @selectionuniform *@selectionstochunif* @selectionroulette @selectiontournament
`StallGenLimit`	Positive integer. The algorithm stops if there is no improvement in the objective function for StallGenLimit consecutive generations	Positive integer | 50, *Inf*
`TimeLimit`	Positive scalar. The algorithm stops after running for TimeLimit seconds.	Positive scalar | *Inf*
`TolCon`	Positive scalar. TolCon is used to determine the feasibility with respect to nonlinear constraints.	Positive scalar | *1e-6*

Table A.1 – Continued

Option	**Description**	**Values**
`TolFcn`	Positive scalar. The algorithm runs until the cumulative change in the fitness function value over StallGenLimit is less than TolFun.	Positive scalar | *1e-6*
`Vectorized`	String specifying whether the computation of the fitness function is vectorized	'on' | 'off'

Appendix B. MATLAB® HILS Architecture Simulation Results

This appendix contains the raw data generated from running a Monte Carlo simulation of 100 trials with the HILS Architecture discussed in ChapterIII. Each row of the table contains the specific chromosome that produced the optimum solution within that iteration.

Table B.1: MATLAB® GA Options Structure [37]

Ro	$\mathbf{R_{max}}$	**JNS**	**f**	$\mathbf{T_w}$
0.4156	1.0684	24.8441	0.1877	1.0375
0.3861	1.0914	24.2786	0.0545	1.3135
0.3861	1.0914	24.2786	0.0545	1.3135
0.1249	0.6047	22.963	1.91	3.1981
0.2665	1.0303	21.8071	1.0003	1.2897
0.2498	0.4743	21.626	0.1248	1.4968
0.3884	0.7877	21.2942	1.8214	1.6941
0.2637	0.7964	19.9649	0.474	0.582
0.1228	1.7789	19.7583	1.9139	1.6561
0.1217	0.5038	19.0000	0.0012	2.5474
0.0622	0.7397	15.1846	1.5414	4.5163
0.2295	0.3669	15.0894	0.6999	0.1955
0.2813	1.0895	14.9566	2.7486	5.3884
0.4389	0.7324	13.8853	0.844	1.9566
0.2529	0.3213	13.2845	0.0352	0.1174
0.2469	0.3306	13.2188	0.1093	0.0837
0.2264	1.4428	13.1866	1.0001	1.2165
0.265	0.3181	12.611	0.0303	0.0532
0.2522	0.2878	12.0304	1.1918	0.0557
0.2296	0.2417	11.7984	0.5	0.0802

Table B.1 – Continued

Ro	$\mathbf{R_{max}}$	**JNS**	**f**	$\mathbf{T_w}$
0.2189	0.712	11.6462	1.0001	3.2875
0.0912	0.3464	11.5861	0.2594	0.2552
0.4608	0.9459	11.2039	1.0478	3.2336
0.496	0.6994	11.1345	0.2427	1.3559
0.0585	0.2342	11.0615	0.2501	0.1763
0.2501	0.5162	9.4077	0.5004	1.7744
0.3739	1.046	7.3049	0.6095	0.6721
0.4989	0.7393	5.3427	0.0982	1.6025
0.2782	0.7256	5.2742	0.5551	0.4474
0.0721	1.8607	5.0721	0.5002	1.7886
0.2489	1.0893	5.0313	1.4743	5.6027
0.0049	0.8416	5.0281	1.0000	5.5777
0.3047	1.1457	5.0038	0.0127	5.6031
0.1498	0.9132	4.6601	0.5000	0.7633
0.0029	1.5805	4.5591	0.2507	1.5776
0.2655	1.2863	4.4641	0.0015	1.0208
0.2502	1.4346	4.2501	0.6444	1.1845
0.0000	0.1797	4.041	0.032	0.1946
0.0064	0.0908	4.0399	0.2681	0.5625
0.0064	0.0908	4.0399	0.2681	0.5625
0.0006	0.1706	4.0163	0.3576	0.17
0.0049	0.0909	4.0015	1.3639	0.5729
0.0041	0.0904	4.0002	1.2347	0.5754
0.0158	1.646	3.9775	1.0000	1.6302
0.0039	0.2319	3.874	1.1909	1.5191
0.1267	1.8481	3.8658	0.1289	1.7213

Table B.1 – Continued

Ro	$\mathbf{R_{max}}$	**JNS**	**f**	$\mathbf{T_w}$
0.2952	0.3237	3.8497	0.0056	0.0285
0.0625	1.2267	3.7578	0.0708	1.1642
0.2248	1.073	3.7572	0.2499	0.8482
0.2261	1.7516	3.708	0.5001	1.5256
0.0171	1.1876	3.5713	0.125	1.1705
0.373	0.5873	3.5275	0.6291	1.4285
0.007	0.0901	3.503	1.585	0.5543
0.4968	0.9731	3.3885	1.3066	3.175
0.1602	0.6641	3.2578	0.2608	0.5039
0.3765	0.7042	3.2534	1.4333	2.1848
0.0123	0.0915	3.236	0.6741	0.5275
0.4776	0.7919	3.2178	0.2500	2.0956
0.2520	0.3400	3.2131	0.2500	0.5867
0.0061	0.1654	3.1933	0.2567	0.1593
0.2518	0.4749	3.1841	1.1056	1.4873
0.4991	0.9991	3.1833	0.000	3.3333
0.3923	0.5029	3.1768	0.1994	0.1106
0.1144	0.1944	3.1568	1.0116	0.5334
0.465	1.2545	3.1433	0.4532	5.2634
0.2895	0.4503	3.126	2.5617	1.0724
0.0051	0.2462	3.1228	0.9687	1.6076
0.0001	0.1773	3.1209	0.0018	0.1772
0.1472	1.2658	3.1051	1.1998	1.1186
0.4275	1.1632	3.0776	0.1875	1.2742
0.4275	1.1632	3.0776	0.1875	1.2742
0.0156	0.2014	3.0723	0.0039	1.2386

Table B.1 – Continued

Ro	$\mathbf{R_{max}}$	**JNS**	**f**	$\mathbf{T_w}$
0.0156	0.2014	3.0723	0.0039	1.2386
0.0295	0.0975	3.0706	0.5814	0.1145
0.4793	1.2513	3.0644	2.5885	5.1469
0.0973	0.6508	3.0469	1.1374	0.9536
0.461	0.6095	3.0377	0.0039	0.2997
0.0184	0.0991	3.0264	0.4514	0.5375
0.064	0.2908	3.0255	0.3525	1.5123
0.0608	0.3514	3.0201	1.827	0.4916
0.5000	0.7069	3.0167	1.0001	1.3793
0.5000	0.7069	3.0167	1.0001	1.3793
0.2415	0.2791	3.0166	2.3286	0.0376
0.0948	0.168	3.0164	1.1674	0.4879
0.4961	0.7055	3.0156	0.002	0.3506
0.4961	0.7055	3.0156	0.002	0.3506
0.0321	0.1182	3.0113	0.043	0.574
0.0087	0.0901	3.0112	0.0206	0.5427
0.0725	0.0958	3.0111	1.4397	0.1554
0.0035	0.0902	3.0079	1.0013	0.5778
0.4301	0.4979	3.0029	1.2934	0.0678
0.4358	1.1698	3.0029	1.3488	0.734
0.4301	0.4979	3.0029	1.2934	0.0678
0.0046	0.1645	3.002	0.7586	0.2845
0.0938	1.8147	3.0002	1.2444	1.7209
0.4999	0.6287	3.0001	0.9296	0.8586
0.1558	0.5368	3.0001	0.0448	2.5402
0.0173	0.0903	3.0000	0.0331	0.4861

Table B.1 – Continued

Ro	$\mathbf{R_{max}}$	**JNS**	**f**	$\mathbf{T_w}$
0.1235	1.736	3.0000	0.5777	1.6125
0.0173	0.0903	3.0000	0.0331	0.4861

Bibliography

1. Nunez, Abel S. et al. "ECM Techniques Generator". *Modeling and Simulation for Military Applications–Proceedings of the SPIE*, volume 6228, 62280Z. 2006.
2. Department of the Air Force. *Electronic Warfare*. AF DD2-5.1. HQ AFDC, Maxwell AFB, AL, November 2002.
3. Schleher, Curtis. *Introduction to Electronic Warfare*. Artech House, 685 Canton Street, Norwood, MA 02062, 1990.
4. Lab-Volt (Quebec) Ltd. *Tracking Radar Model 9625 Instruction Manual*. Telecommunications Radar, May 2000.
5. Hong, Seng, Michael A. Saville, Chad Simpson, Patrick Marshall . "Investigation on Genetic Algorithm for Countermeasure Technique Generator". *Signals, Systems and Electronics, 2007. ISSSE '07. International Symposium on*, 351–354. ISSSE, Montreal, QC, Canada,, 30 July-2 August 2007.
6. Mayhew, Oscar. *RADAR System Characterization Extended to Hardware-in-the-Loop Simulation for the Lab-Volt™ Training System*. MS Thesis, AFIT/GE/ENG/07-29. School of Electrical and Computer Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 2007.
7. Skolnik, Merrill L. *Radar Handbook*. McGraw-Hill, 1221 Avenue of the Americas, New York, NY, 2nd edition, 1990.
8. Balanas, Constantine A. *Antenna Theory*. John Wiley and Sons, Inc., 111 River Street, Hoboken, NY, 07030, 3rd edition, 2005.
9. Skolnik, Merrill L. *Introduction to Radar Systems*. McGraw-Hill, 1221 Avenue of the Americas, New York, NY, 3rd edition, 1988.
10. Blackman, Samuel S. *Multiple-Target Tracking with Radar Application*. Artech House, 685 Canton St., Norwood, MA 02062, 1986.
11. Zhang, Yan and Ram M. Narayanan. "Design Considerations for a Real-Time Random-Noise Tracking Radar". *IEEE Transactions on Aerospace and Electronic Systems*, 40(2):434–443, April 2004.
12. Back, Thomas, David B. Fogel, and Zbigniew Michalewicz (editor). *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Ltd., The Public Ledger Building, Suite 1035, 150 South Independence Mall West, Philadelphia, PA 19106, 2000.
13. Haupt, Randy L. "An Introduction to Genetic Algorithms for Electromagnetics". *IEEE Antennas and Propogation Magazine*, 37(2):7–14, April 1995.

14. Coello-Coello, Carlos A., David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for solving Multi-Objective Problems*. Kluwer Academic Publishers, 233 Spring Street, New York, NY 10013, 2002.

15. Landis, Nathan and Dr. Gary Lamont. *ECM/GA Techniques Generator*. Quarterly report, AFIT/AFOSR/AFRL, Bldg 641, Rm 307B, 2950 Hobson Way, WPAFB, Dayton, OH 45433, Winter 2007.

16. Johnson, J. Michael and Yahya Rahmat-Samii. "Genetic Algorithms in Engineering Electromagnetics". *IEEE Antennas and Propagation Magazine*, 39(4):7–22, August 1997.

17. Back, Thomas, David B. Fogel, and Zbigniew Michalewicz (editor). *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing, Ltd., The Public Ledger Building, Suite 1035, 150 South Independence Mall West, Philadelphia, PA 19106, 2000.

18. Golino, Giovanni. "Improved Genetic Algorithm for the Design of the Optimal Antenna Division in Sub-Arrays: a Multi-Objective Genetic Algorithm". *Radar Conference, 2005 IEEE International*, 629–634. March 2005.

19. Kajenski, Peter J. "Automated Optimization of Tracking Algorithms Using Simulated Annealing". *Radar Conference, 2007, Proceedings of the IEEE*, 177–179. Waltham, MA, 17-20 April 2007.

20. Wang, Jianag, Weiting Liu, Min Wang, Bing Zhang, and Jian Wang. "Multiple Maneuvering Target Data Association Based On Genetic Algorithms". *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on*, volume 2, 1011–1014. February 2005.

21. Townsend, James, Michael A. Saville, Seng Hong, Chad Simpson, and Oscar Mayhew . "Waveform Optimization for Electronic Countermeasure Technique Generation". in–press. Rome, Italy, 2008.

22. Saville, Michael A. "Personal notes on modeling the Lab-Volt™ Radar", December 2007.

23. Lab-Volt (Quebec) Ltd. *Radar Jamming Pod Model 9608-10 Instruction Manual*. Telecommunications Radar, May 2000.

24. Cheney, Margaret. "A Mathematical Tutorial on Synthetic Aperture Radar". *Society for Industrial and Applied Mathematics Review*, 43, No. 2:301–312, 2001. S0036144500368859.

25. Schleher, Curtis. *Electronic Warfare in the Information Age*. Artech House, 685 Canton Street, Norwood, MA 02062, 1999.

26. Chrzanowski, Edward J. *Active Radar Electronic Countermeasures*. Artech House, 685 Canton Street, Norwood, MA 02062, 1990.

27. Boyd, Joseph A., Donald B. Harris, Donald D. King, and H. W. Welch Jr. *Electronic Countermeasures*. Peninsula Publishing, Los Altos, California, 1978.

28. Tipler, Paul A. *Physics*, volume 1: Mechanics, Oscillations and Waves, Thermodynamics. W. H. Freeman and Company, 41 Madison Avenue, New York, NY 10010, fourth edition, 1999.

29. Lab-Volt (Quebec) Ltd. *Principles of Radar Systems Student Manual 28923-00*. Telecommunications Radar, May 2000.

30. DiFilippo, D., G. Geling, and G. Currie. "Simulator for Advanced Fighter Radar EPM Development". *IEE Proceeding of Radar and Sonar Navigation*, volume 148, no. 3, 139–146. June 2001.

31. Ward, James. *Space-Time Adaptive Processing for Airborne Radar*. Technical Report 1015, MIT Lincoln Laboratory, Lexington, MA, USA, December 1994.

32. Sarkar, Tapan K. and Sheeyun Park. "A Blind Least–Squares Approach to STAP using MCARM Data". *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems and Computers, 1998*, volume 2, 1552 – 1556. Pacific Grove, CA, November 1998.

33. Belkacemi, Hocine and Sylvie Marcos. "Fast Iterative Subspace Algorithms for Airborne STAP Radar". *EURASIP Journal on Applied Signal Processing*, 2006:Article ID 37296, 8 pages, 2006. Doi:10.1155/ASP/2006/37296.

34. Townsend, James, Michael A. Saville, Richard A. Martin, Seng Hong. "Simulator for Velocity Gate Pull-Off Electronic Countermeasures Techniques". *Radar Conference, 2008. International*, in–press. Rome, Italy, 2008.

35. Guerci, J. R. *Space-Time Adaptive Processing for Radar*. Artech House, 685 Canton Street, Norwood, MA 02062, 2003.

36. Varberg, Dale and Edwin J. Purcell. *Calculus*. Prentice Hall, Upper Saddle River, NJ 07458, 7th edition, 1997.

37. The MathWorks Inc., 21 Elliot St. South Natick, MA, 01760. *Genetic Algorithm and Direct Search Toolbox*.

38. Strang, Gilbert. *Linear Algebra and Its Applications*. Thomson Learning, Inc., Belmont, CA 94002-3098, 4th edition, 2006.

Vita

James D. Townsend was born in Bangor, Maine in 1981. After graduating from John Bapst Memorial High School in 1999, James attended Worcester Polytechnical Institute, where he was a member of Reserve Officer Training Corps Detachment 340. During his time in Detachment 340, James served as Wing Commander, running day-to-day operations for over 60 cadets. James was also inducted as a member of Eta Kappa Nu in 2002 and is still an active member. James received a Bachelor's of Science Degree in Electrical Engineering from Worcester Polytechnical Institute in May of 2003. After being commissioned in the United States Air Force in May of 2003, his first assignment was to the National Air and Space Intelligence Center (NASIC) at Wright-Patterson Air Force Base, OH. While at NASIC, James worked in the Data Exploitation Directorate, Signals Exploitation Division as a Airborne Weapons Analyst, producing timely reports on current and future threats. James enrolled at the Air Force Institute of Technology in August of 2006 as a Master of Science in Electrical Engineering student. He completed the RADAR sequence with research emphasis in advanced RADAR design and signal processing. Following graduation from the Air Force Institute of Technology, James will report to the 453rd Electronic Warfare Squadron at Lackland Air Force Base, Texas.

Permanent address: 2950 Hobson Way
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

Index

The index is conceptual and does not designate every occurrence of a keyword. Page numbers in bold represent concept definition or introduction.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
3-3-2008	**Masters Thesis**	September 2006-March 2008

4. TITLE AND SUBTITLE

Improvement of ECM Techniques through Implementation of a Genetic Algorithm

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Townsend, James D, Captain, USAF

5d. PROJECT NUMBER
JON 07-334

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765 DSN: 785-3636

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/08-34

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr. Seng M. Hong, Air Force Research Laboratory (MAJCOM: AFMC)
RF Countermeasure Assessment Lab (RYRA)
2241 Avionics Circle
Wright-Patterson AFB OH, 45437 DSN: 904-9636

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RYRA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This research effort develops the necessary interfaces between the radar signal processing components and an optimization routine, such as genetic algorithms, to develop Electronic Countermeasure (ECM) waveforms under a Hardware-in-the-Loop (HILS) architecture. The various ECM waveforms are stored in an ECM library, where an operator selects the desired function to use with a particular system. This optimization works with modular components, compared to previous research that embedded a genetic algorithm into the Range Gate Pull-off (RGPO) waveform optimization loop, which can be interchanged based upon the operator's desired hardware/software testing setup. The ECM library's first entries contain the RGPO and Velocity Gate Pull-off (VGPO) signals, developed mathematically for multiple polynomial profiles representing realistic moving false targets. The Lab-Volt training system and jammer pod provided a validation medium for the developed RGPO and VGPO waveforms. These waveforms were optimized using a Simulink model of the Lab-Volt radar system and the MATLAB Genetic Algorithm (GA) and Direct Search toolbox, contained in Version 7.4 (R2007a), using a defined parameter set, specified for the RGPO waveform. Integration of MATLAB code with Simulink models provides the necessary interfaces to transition from software radar models to actual system hardware. Results from GA optimization illuminate the necessity to specifically define the necessary constrains, both linear and nonlinear, imposed upon the environmental conditions. Given defined constraints relative to the Lab-Volt training system, the HILS architecture produced multiple constant velocity range profiles with walk-off ranges and maximum velocities similar to the Lab-Volt Jammer Pod.

15. SUBJECT TERMS
HILS Optimization, technique generation, countermeasures, genetic algorithm

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			**Maj. Michael A. Saville**
U	**U**	**U**	**UU**	134	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565x4719; email: michael.saville@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18